APPENDIX I AND II (To STAP paper on Options for SAFE

DIRECTOR OF CENTRAL INTELLIGENCE
Science and Technology Advisory Panel

18 MAR 1980

MEMORANDUM FOR: Director of Central Intelligence
Deputy Director of Central Intelligence

STATINTL FROM: [redacted]
STAP Chairman

SUBJECT: Questions Regarding SAFE

1. In response to your request of March 14, 1980, for comments on the current status of SAFE, I attach a list of eight questions and brief comments that STAP believes should be addressed.

2. STAP is continuing its analysis of the SAFE problem and will prepare an options paper within the next two weeks for your consideration.

STATINTL [redacted]

Attachments:
As Stated

1.0 What steps are being taken to ensure that the Agency, rather than the contractor is in control of the technical aspects of the design of the system?

The geographic remoteness of the contractor, and the lack of continuing contractor-user interaction can lead to a situation in which the builder of the system also becomes the architect. The absence of reciprocal technical representation, like resident engineers, delays every routine decision and makes larger ones unresponsive to Agency/community needs or technological constraints. Strong Agency technical management is absolutely essential if the system is to satisfy real and evolving agency needs and if it is to be integrated with other Agency and community resources.

2.0 How is SAFE management ensuring that a final working system has been developed from the continuing evolution of an operationally valid pilot system? How has SAFE taken advantage of the experience of similar, very large systems in their:

1) system architecture,
2) communication and control, and
3) changing performance requirements?

How is SAFE management ensuring that the system will:

1) make available data on operation and usage of the pilot system to guide development;
2) be able to modify both system functions and interaction capabilities so as to meet changing and evolving requirements; and
3) be able to add new functions and interactions so as to meet new requirements?

Relevant Experience -- It is well agreed that an information handling system cannot be achieved by a simple Design-Build-Use cycle, no matter how brillant the design or faithful the building. Consider three (out of many) currently operating very large nets with requirements at least comparable in size and complexity with the Agency/ community's:

1) the ARPA net,
2) the airlines reservation system, and
3) the IBM in-house computing net.

All of these systems evolved - that is, they started as soon as possible with operating pilot systems, so that there was always an operational evaluation of effectiveness. Furthermore, in each case, a primary purpose of the original plans was drastically altered as experience was gained. Even so, developments in concept, hardware, software, and practice are continuing now in greater volume than ever.

2.1 The ARPA Net

The ARPA net was originally conceived by [redacted] STATINTL at the ARPA IPO (Information Processing Office) as a means of netting research computers in order to do distributed computing. The message facility was then a minor function. As message usage rose, packet switching became a powerful tool with wide application elsewhere in the technology. The separate centers in the ARPA net have much independence, subject to some fairly strict requirements for communications protocols and access. This allows competitive development and a common evaluation of new technological developments - e.g., the intelligent terminal is being subjected to widespread experimentation and development.

2.2 Airlines Reservation System

The first automated electronic airfare reservations and ticketing systems were disasters - e.g., SABER, the American Airlines/IBM effort was predesigned and built to an apparently reasonable set of specifications that turned out not at all to match the operational needs. The first successful ones were ad hoc temporary devices (e.g., UNITED) that worked just well enough to be improved.

Note that the requirement for fast and accurate inter-line communications, backed up by automatic commitments for seats and equipment was a much later development; it is, however, by now one of the most valuable and cost-effective facilities.

2.3 The IBM In-House Computing Net

Virtually all the on-line computers at IBM company installations world-wide are netted together by communication

facilities, making them, we believe, the largest net in the world. Admirable new functions exhibited by their systems are:

1) a modifiable macro command language,
2) a consistent, speedy and flexible data transmission/translation scheme.

3.0 The SAFE user community consists of Intelligence Community analysts covering the full spectrum of research into foreign political, military, economic, scientific, and technological activity. Their effective use of this system and, ultimately, the quality of intelligence they produce rest on whether their real needs can be identified and satisfied by the system. To that end, what actions will be taken to ensure that:

1) all elements of the intended user community are actually involved in the system's continuing development,

2) the broadest of the present analytical requirements are identified,

3) these present requirements can be validated by a consistent method,

4) the validated present requirements will be met,

5) modifications and new requirements can be accepted as they are identified,

6) all analytical users will acquire the necessary skills and familiarization with SAFE on an interim basis so they are ready to begin broad utilization when system IOC is reached,

7) this interim SAFE test-phase acquires continuing comprehensive experimental data on user experience with the system, and

8) the acquired data on user experience is actually utilized in the architecture and development of the system?

4.0 How can the Agency make a reasonable evaluation of the current status of SAFE with major portions of the proposed operational capabilities either unspecified or uncommunicated to the Agency? For example,

1). the user command language and its parsing,
2) the user programming languages,
3) the user editing languages, and
4) procedures for backup, including regeneration of derived files lost in crashes.

Note the above have to be prototypes capable of continuing responsive evolution, rather than final imposed prescriptions.

5.0 What actions are under way to insure that the Intelligence Community has access to CIA SAFE and that CIA SAFE has access to DIA SAFE as well as such systems as COINS and SOLIS?

The absence of appropriate linkages with other IC systems makes it highly probable that duplicate facilities and files will be acquired and constructed with higher costs and lessened capability for the total IC system. Provisions for such linkages should be built into SAFE from the start, otherwise it will be difficult if not impossible to backfit these linkages.

STATSPEC

6.0 How will SAFE deal with open source material? Will [redacted] material, either finished publications or field reporting, be made available to the analyst through SAFE? How will SAFE deal with current newspaper and journal entries?

Several of the offices that will use SAFE, in particular OPA and OSWR, make extensive use of open source material. Their analytical efforts will be seriously hampered if their files do not include open source materials.

7.0 What steps are being taken to ensure that SAFE will be designed to allow collaborative usage?

Examination of other similar systems, such as Stanford University's SUMEX system or the internal IBM system, shows

that user-to-user interactions comprise a significant fraction of the total use of these systems and greatly enhance the overall analytical capability of its users.

8.0 What steps are under way to ensure that in the procurement of major hardware items, these items will be compatible with existing Agency systems?

The Agency has made very substantial investment in ADP equipment that currently serves a wide variety of users. This investment should be capitalized on in order to enhance the future capabilities and particularly the flexibility of SAFE. If the interoperability of SAFE and existing ODP hardware is going to be dependent on software, then provisions should be made:

1) for the development of the needed software since it will be a major undertaking; and

2) for the establishment of evolving standards and protocols for interconnection.

APPENDIX II

MANAGING VM/CMS SYSTEMS FOR USER EFFECTIVENESS

Walter J. Doherty and Richard P. Kelisky, IBM Thomas J. Watson Research Center,

Yorktown Heights, New York, USA

Abstract

Management of the VM/CMS systems at IBM's Thomas J. Watson Research Center is based on assumptions that the user's time and work product are valuable, and the system being used should be managed so that the user's ability to work is enhanced with least inconvenience to the user. Within the financial limits on Computing Center hardware and personnel it is possible to provide user-effective interactive computing services which approach these objectives by placing highest priority on system reliability and availability followed by performance improvements and introduction of new functions. Because our studies show that user productivity is well correlated with system response we are convinced, first, that it is essential to maintain a management commitment to the goal of good service to the laboratory, and second, that published service schedules must be adhered to. At the same time, it is necessary to realize that users will have special problems, and management must be willing to work out solutions for these users. The goal of the computer operator is to keep the system up, and system programmers must introduce necessary system changes in a way which does not jeopardize stability. We realize that the computing system is a tool which enhances both the user's memory and reasoning power. We have, therefore, assumed responsibility for preserving our users' data. We have also developed data migration mechanisms to aid the user in managing data.

Introduction

This paper describes the evolution of VM/CMS interactive computing services at the IBM Thomas J. Watson Research Center, and the necessary changes in viewpoint needed to manage interactive computing services effectively for the user community. Benefits deriving from alternative strategies have been evaluated against the following criteria:

(1) The computer user's time and work product are valuable. Therefore, the interactive systems on which the user works should be managed so that his ability to work is enhanced with least inconvenience to him.

In fact, for most interactive computing the aggregate user time is more costly than the computer time. Figure 1 shows the cumulative percentage of people using computers at Yorktown *vs.* the cumulative CPU cycles consumed by those people. Similar curves characterize the distribution of computing usage in many other installations. (1) In general, about five percent of the people consume about seventy five percent of the computing resource. If we focus on the top five percent of the computer users we find that a small fraction of their interactions dominates their demand for computing. The computer user's technical management, not the Computing Center, must determine whether or not the machine intensive computing to be done is technically justified. Machine intensive computing occurs for about 5 percent or less of the interactions. The remainder of the interactions involve immediate, ongoing communications between people and the computer. If we use a figure of $800 per hour for the cost of a large computer today, the 25 percent used for 95 percent of the interactions costs $200 per hour. One hundred simultaneous users, whose own time might be worth at least $20 per hour, cost $2,000 per hour. These figures are only for illustration. In fact, our computer costs are less, our users' time is worth more, and we normally have many more than 100 simultaneous users on each of two VM/370 systems at Yorktown. Therefore, for more than 95 percent of the computer interactions, the user's time is much more costly than the computer's. Furthermore, these costs are diverging. The Computing

Center at IBM's Thomas J. Watson Research Center attempts to provide an environment in which the computer user can then make effective use of his time, and efficient use of the computing resources provided.

Figure 2 shows the growth in the interactive use of computing at Yorktown from 1973 through 1977. It is clear that the aggregate cost of the users' time is greater than our computer costs, and this difference continues to grow. Therefore, the Computing Center seeks maintenance and enhancement strategies which, within budget limits, help the user to get work done and which tend to give highest priority to the user's requirements for service. This means that in so far as possible we must isolate changes from each other and allow both old and new versions of functions to coexist so that the user is not forced to change his ways of working.

An essential tool for our maintenance and enhancement strategy is the virtual machine in VM/370. In virtual machines we are able to test new versions of most subsystems or applications without disturbing existing versions. We can isolate many special functions to separate virtual machines which gives us the ability to tailor the appearance of that function and simplify the user interface without fear of "contamination" from other non-related function. When CMS is the operating system in the virtual machine, we find that we often have least system overhead for performing that function.

(2) Complexity in the man-machine interface is wasteful of users' time, but complexity can be reduced by the judicious use of procedures (2).

In April, 1978, the Yorktown Computing Center processed more than 10 million interactions. An interaction is any user input and its accompanying system response. If the information content per interaction is too small, people could be decreasing the rate at which they work rather than increasing it. Or, viewed another way if we can decrease the amount of work the user has to do to derive a useful result from an interaction, then we have made the system more effective for that user.

We find that often our computer users will take great care to determine how to use some combination of computing functions for a task performed repetitively. Once they have done this the combination of functions is given a name and stored on secondary storage. After the computer has stored this information, the person need not repeat those thought processes again. These procedures (EXEC's) may include useful default values for parameters as well as logic to make the computer adapt to the user's environment, rather than requiring the person to adapt to the machine. Once a user has constructed an EXEC, others can, and do take advantage of it. For example, EXEC's exist to simplify the business of using IBM's internal computer network. The user normally doesn't have to concern himself with the kind of data involved, the protocols, or the ways of addressing the person or group to whom data is sent or from whom it is received. Good default actions have been determined by one of our users, and the EXEC's which he developed are now routinely used by most of our network users at Yorktown. The naive user need not even be aware of the existence of the commands or parameters inside these EXEC's. By this process complexity is reduced, typing errors are avoided, and the user-effectiveness of the interaction increases.

From examining twenty percent of the users on one of the VM/CMS systems at Yorktown, we find that there are twice as many EXEC's as all other source programs in conventional programming languages combined. (3) In 1977, the Stanford Linear Accelerator Center (SLAC) installation observed an average of twenty five commands executed per command typed at a terminal (4), when using the Wylbur interface. Wylbur is an editing interface for interactive computing. Wylbur was about ten years old at that time. In 1971, we found at Yorktown that the number of commands executed per person, counted at execution time after procedure expansion, doubled in just five months. Thus, our experience indicates that there is continued growth in function as users learn to adapt the system to their requirements. We can think of this growth as arising from *captured intelligence*. In the early 60's the computing industry speculated on the notion of *artificial intelligence* whereby the computer would heuristically determine the solution to some problem. In practice, we find the concept of *captured intelligence* to be extremely useful for raising the man-machine interface to a meaningful level for many people.

(3) Display terminals are able to provide a user more information in a given time than typewriters. They can be especially valuable to the user as an aid to how to use the system.

Where justified by the user's work in the view of the user's management, it is our objective to place an IBM 3277 display terminal on each user's desk. In general, we find terminal usage of an hour or more per day is a reasonable prerequisite to installation of a terminal in a user's office. An example of the value of the display terminal in accelerating the user's work is provided by our documentation practices on EXEC's. It has become customary, at Yorktown, to include comments on how to use any EXEC directly inline with the EXEC code. Thus, each user can determine what functions the EXEC does for him, what its syntax is, and what the default values of parameters are by simply typing the name of the EXEC followed by a "?". By having the comments included inline with the EXEC, these comments are usually changed whenever the EXEC is changed, which maintains currency between the documentation and the code, and they are there whenever needed. Our Yorktown user community has told us that these "self-defining" commands are regarded very much more favorably than all other forms of documentation or education. This facility was simply not feasible with the slower data rates of typewriter terminals.

(4) System programming is costly; therefore, management seeks solutions to user problems which do not lead to growing commitments in system programming time.

System programmers are a scarce and costly resource. Strategies which increase system programming requirements are to be avoided. In general, a local change to an operating system carries with it an obligation to reevaluate that change everytime there is another change to that part of the operating system, e.g., PLC, SU, PTF, etc. Such local changes, then, represent a kind of promissory note on which one pays interest (system programming time) indefinitely. The Yorktown Computing Center seeks strategies to reduce this cost. For example, the Computing Center selected a strategy to attach new devices which avoids repeated reprogramming as the underlying operating system changes. This subsystem, called the Advanced Terminal Subsystem, attaches non-IBM terminals to an IBM System/7 which has been programmed so that the

underlying operating systems (VM/CMS, MVS/TSO, and formerly, TSS) on the 370/168 are unchanged. Change is localized to the System/7, and new terminal types are introduced by means of table entry rather than by modifications to MVS or VM.

We have also distributed the responsibility for change to the people who most need it. By giving them special virtual machines, and access to whatever source code they needed, together with good tools for debugging changes and for finding out who else might be interested in any given function, we provide an environment in which changes take place but reliability remains high. Special libraries exist to allow users to make their changes available to others. By simply typing the command OWNER followed by a command name, any person can determine who the current owner of that version of the function is, and communicate directly with that person through the system for describing problems or new ideas for enhancement. Once such functions have proven to have broad value to many people, they can readily be incorporated into the central Library with little effort by the systems programmers.

(5) The development of new computing services is costly; therefore installation management develops new services incrementally in order to evaluate these services before the next step is taken.

We avoid large development projects which must span several years before their benefits are available to the users. Not only are such projects costly, but there is the great risk that the problem to be solved will change or disappear during the lengthy development process. Because program development is a very complex process, we find that feedback is required continually throughout the development cycle. By bringing the computer to the person who has the greatest knowledge of the problem to be solved, and providing that person with adequate computing resource and an appropriate set of tools, we find that he can build a prototype solution, try it personally, get others to try it, and iterate many times in this fashion until the prototype is perfected to yield the required function. We find that this is the fastest way to produce high quality function for the least cost. We also find that very skillful programmers absolutely require such feedback throughout the development cycle. There is accumulated evidence to show that most errors have

occurred in the design stage (5), and that individual skills are the major factor in determining the quality of programming projects of less than 50,000 lines of executable source code in size.

By developing new services in this iterative fashion, we are more likely to be sure that we are solving the right problem, that the perceived solution is indeed a solution, and that the function can be readily changed as future needs dictate. VM/CMS and the virtual machine available under VM give us the basic environment we need to work in this fashion. Our experience provides evidence that this iterative method of program development is an improvement over the traditional programming development process.

(6) User need for asynchronous computing grows proportionally to interactive computing; therefore the availability of well-planned distributed function is increasingly important.

The load distribution curve of Figure 1 is the same for interactive computing as it is for traditional batch processing. Because of the growth that has continued for the past ten years in interactive computing, there is pressure to off-load that part of people's work which is large in computing resource demand, or which has reached some logical stage of completion. If this were not done, long running processes would act as a block at the terminal preventing a person from accessing his or her other data while that long running process completed. By packaging special functions in special purpose virtual machines, it becomes easier to off-load work whenever appropriate. These special purpose virtual machines need not run on the same real machine, and in many cases, do not. However, to facilitate communication between such special purpose virtual machines, A. N. Chandra developed an experimental mechanism which greatly facilitates communication between different virtual machines running on the same real machine. We have employed this in our text processing work, our laboratory automation work (6), our MSS data migration work, etc. The Virtual Machine Communication Facility (VMCF) portion of VM/CMS is an outgrowth of Chandra's work. The Network Job Entry and Network Job Interface software products have given us the necessary communication facilities between different real machines. By being able to communicate with other machines, whether virtual or real, we can permit new and old functions to coexist. Our users *communicate with old function as needed*

rather than convert old functions to new environments. By insisting on having most of our terminals locally attached, we minimize response time delays, reduce security problems associated with remote access, and reduce the number of separate components required to be available simultaneously. Thus we find that *networking data rather than people* is a more effective way to promote collaboration and avoid replication of work.

In the following sections, management actions will be discussed as they affected the Growth of Interactive Computing, Availability and Reliability, Response Time, Expansion Factors, and Visibility of Service, Data Management, and Distributed Function.

The Growth of Interactive Computing at the Watson Research Center

The computer user at the IBM Thomas J. Watson Research Center has available a variety of computing services. VM/370 is provided on two (six megabyte and seven megabyte) 370/168's and OS/VS2 release 3.7 (MVS) together with TSO is provided on an eight megabyte 370/168-3. With their manager's concurrence, employees at the laboratory are urged to employ these computing facilities in order to do creative and innovative work. Usage of the computer is not restricted to the scientist; secretaries and other non-technical people use computers if management determines that it is cost-effective for them to do so. Since nearly all computing services are accessed interactively from terminals in user's offices, terminal rooms or, in some cases home terminals, we have attempted to develop management policies and practices which make interactive computing services effective for the user. Some of these policies will be described in the following sections. In what follows, prime shift means the period 08:00 to 18:00 Monday through Friday excluding holidays. Second shift is 18:00 to 24:00 Monday through Friday and 08:00 to 18:00 Saturday, Sunday, and holidays. Third shift is all remaining time.

Interactive computing began, as a service, at the Thomas J. Watson Research Center with the introduction of APL in 1965. To the computer user APL presented a fundamentally new interface different from the

interface of batch computing. Clearly, most of the strong user acceptance of APL resulted from the APL language and the APL system, but these positive characteristics might not have sufficed for user acceptance if at the same time the APL group had not recognized that this new way of computing required a new way of managing computing services. Simply stated, they recognized that interactive computing must be *available* in order to be useful. The user must be able to turn on his terminal at any time he is ready to work and, with a high degree of confidence, find the interactive system ready to serve him. APL service was provided on an IBM 360/50 essentially twenty-four hours a day, seven days a week. Preventive maintenance of the 360/50 was limited to a few hours every three weeks, and because of the intrinsic reliability of the APL system and the machine, the computer user was able to assume that APL would be available whenever he was ready to work: in the evening, in the middle of the night, on weekends, holidays, etc. While Computing Center management may not have realized it at the time, criteria for user effectiveness, system availability and system reliability were being established for interactive computing services. These criteria have strongly influenced subsequent management practices of the Watson Research Computing Center.

In 1967, TSS/360 was introduced at the Watson Research Center on an IBM 360/67. On TSS we began to experiment with and to understand the effect on user productivity of variations in response time, the value of indicating to the user a measure of system load *prior* to his logging on, the importance of scheduling controls in order to distribute system resources equitably, and the need for "transparent" management of a growing on-line user data base. TSS has been terminated at the Watson Research Center, but the experience gained from TSS suggested new extensions to VM/370 and TSO so that these systems might become more effective for the user.

In 1969, CP/67, the predecessor of VM/370 was introduced on a second 360/67 to serve as a software development facility for operating systems. The irresistible user demand for CMS, which was not offered at first to the computer users, showed the importance of providing an easy-to-use system for the non-specialist. The increased load on the system resulting from demand for CMS showed that it was necessary to give the

programmers, who had responsibility to maintain CP/67 (and later VM/370 which replace CP/67 in 1972) tools to enable them to understand what is taking place inside the system so that a few users could not usurp system resources to the detriment of system performance for the entire user community. In particular, we learned that interactive system users are reluctant to accept the loss of a useful function or facility which they used on an earlier interactive system, e.g., something equivalent to the ease of data management provided on TSS must be made available on VM/370.

In the following sections we will examine in more detail results of our experience with managing interactive systems.

Availability and Reliability

Computing services at the Watson Research Center are available, insofar as possible, twenty-four hours a days and seven days a week. Clearly, one must ask if such a schedule is necessary. Does the Computing Center have cost-justified requirements from computer users for such a schedule? Technical management has authorized computing which requires more than prime shift, but there are other reasons why such a schedule was established: the requirement for computer availability by the experimentalists, the necessity that system maintenance be done after prime shift, and the relatively low incremental cost to provide these added services. There are, however, impediments to such a schedule: unscheduled software or hardware failures, scheduled software changes or tests, scheduled hardware shutdowns for engineering changes, preventive hardware maintenance, hardware relocation, and finally, the unavailability of personnel to operate the systems on weekends and holidays.

On the question whether such extreme availability is necessary we note that researchers (computer scientists, chemists, physicists, engineers, etc.) expect and require a degree of flexibility in working conditions. Experiments cannot be constrained to the normal eight hour working day, and there are experimentalists who may extend or shift their working hours to coincide with the requirements of their experiments. There are

also scientists who work more than eight hours, or are simply more productive if given access to the tools they need in the middle of the night, on weekends and on holidays. Finally, we realized that if users are to entrust all of their programs, procedures, documents, and data to the interactive systems then that information must be available at least in the same way as from a locked file in the user's office. To limit the user's access to his files is simply not satisfactory from his point of view. Consequently, Computing Center management made the decision to offer computer services, insofar as possible, around the clock. This policy is at times incompatible with the needs of the Computing Center to test new software at night, modify or repair hardware or carry out necessary housekeeping tasks such as making copies of the on-line data base as a safeguard against destruction. But, given a commitment to provide service around the clock, compromises can be worked out so that many hardware changes are postponed to third shift or to weekends. Together with our users we have evolved a strategy for software testing so that changes to the software system are introduced on pre-scheduled days, usually during a shutdown at 18:00 lasting 15 to 20 minutes. The period from 18:00 to 08:00 the next day is then designated as a User Risk Session which informs our users that new and possibly unstable software is being tested. The Computing Center is then allowed two system crashes. After the second crash, the standard prime shift system is restarted, and the users know that no more testing will be done that night. Housekeeping tasks can usually be designed to run in a time-shared mode so that back-up of the data base can be done after 18:00 even though users are on the systems. Such tasks take longer to complete when executed time shared rather than stand-alone, but stand-alone use of the computer is employed very rarely because it limits user access to the systems.

When new software changes to the operating systems have been tested for about a month on second and third shifts, they are then introduced on prime shift, usually on a Friday so that we have the weekend to solve new problems which may appear. In general, after one prime shift failure of the new software we return to the older standard system. If the new software runs for a week without prime shift failures it becomes the new standard system.

Unscheduled hardware failures normally call for immediate repair in many computing centers. In general, our policy is to restart the system and maintain service even if it is degraded. Only if the failures become so frequent that the user cannot proceed with his work, or if we cannot restart the system, will we give up the system for service on prime shift. Relocation of hardware and the addition of engineering changes is constrained to second and third shift periods or on weekends, excluding 08:00 to 18:00 on Saturday if possible.

Despite our intentions to provide around the clock service, a major difficulty is that of operator availability. Because we always operate under manpower limitations, we were led to seek new ways to increase operator availability with a fixed number of computer operators. We realized that we could not operate each machine from its own console and continue to mount tapes and disks on demand with the operations staff we have. We, therefore, designed a central "operations bridge" from which all three 370/168's could be operated by means of IBM 7412's and 3277's, and made changes to the operating system where necessary so that systems could be started and controlled remotely. Since all of the operaters must be trained to operate all of the systems, it is much more difficult to develop operators highly skilled on all systems. Consequently, we have an MVS/TSO/JES3 operator team with a lead operator and a VM/370 operator team with its lead operator. Members of one team can operate the other system, but do not necessarily have specialized knowledge of their secondary system. This strategy does allow the Computing Center to operate on weekdays with six operators on prime shift, five operators on second shift and three operators on third shift. On weekends and holidays there is one operator on each of two 12 hour periods per day. On certain holidays, such as Christmas, New Year's Day and special IBM work holidays the interactive systems run unattended with the understanding that private tapes and disks cannot be mounted. Nevertheless, there are people who come into the laboratory on holidays to use the computers, or who, in a limited number of cases, dial in from home terminals which represent yet another way to extend system availability.

Another important aspect of system availablility is that of management flexibility. Computing Center management must be willing to make compromises in schedules in order to serve special needs of users. When Computing Center schedules are published each week, or when sudden changes to those schedules must be published by means of logon messages to the users, we invite users with special needs to contact us. If a user is attempting to finish a paper to meet a deadline, or simply needs to get an urgent piece of work done by a certain time, it is important that the Computing Center try to adapt its needs to the user rather than insisting that users always adapt to the Computing Center. On the other hand, it is important that schedules promising service at specified hours be adhered to. It is extremely frustrating for a computer user to come into the laboratory on a weekend or holiday expecting to use the computer and find that the Computing Center has changed the schedule even though the reasons may be good ones. We have also established a special telephone number from which the user can obtain a recorded message on the status of the systems. This message is changed and time-stamped whenever a system failure or other emergency changes the service schedule. The main disadvantage of the recorded message is that it is just not accurate enough for brief system failures although it is useful for failures lasting more than five minutes. Our users tell us it is accurate no more than 70 percent of the time, and we are seeking a better way of informing users of the status of the systems.

Performance Management and Visibility of Service

System response time is the time measured from the user's signal to the system that there is work to be done to the point at which the system begins to present those results to the user. It is time the user had to wait for results and is important to his ability to continue effectively with his work. Figure 3 shows a picture of the response time to a single small program which ran every 20 seconds in the spring of 1971. Each 20 seconds, it awakened, and recorded the number of logged on users and the response time that it received from the system. Figure 3 is really a 3 dimensional graph. Each plot position is a number indicating the number of times a specific response time occurred.

Notice that as the number of simultaneous users increases, the response time steadily and sharply degrades. Thus, if we simply try to maximize the number of simultaneous users, we find that we are maximizing the number of people to whom we are presenting a sharply degraded picture of service. This means that the number of simultaneous system users is a misleading measure of system load. The *system expansion factor* rather than the number of simultaneous users is a meaningful measure to the Computing Center and the users. It is defined to be the ratio of the actual time to do a unit of computer-limited work to the minimum time to do that work in a stand-alone environment. We will discuss the expansion factor in more detail below.

We find that significant performance improvements lead to a reduction in the number of simultaneous system users, perhaps because the user gets done with his planned work sooner. This is accompanied by an increase in the number of interactions processed, but the overhead needed to do that work tends to decrease.

Figure 4 shows the profound influence that system response time degradation can have on user behavior. If we break an interaction into two parts, the system response time (SRT) during which the system is processing a request for the user, and a user response time (URT) during which a user is keying-in his or her next request to the system; then each second of system response degradation leads to a similar degradation added to the user's time for the following request.

This phenomenon seems to be related to an individual's attention span. The traditional model of a person thinking after each system response appears to be inaccurate. Instead, people seem to have a sequence of actions in mind contained in a short term memory buffer . Increases in SRT seem to disrupt their thought processes, and this may result in a time consuming reloading or alteration of the short term memory buffer.

This phenomenon was first discovered by S.J. Boies (7) at Yorktown in 1971 while doing normative studies of interactive computing. We had recorded every interaction on TSS for three years, and this data base included a wide range of human tasks. Boies' phenomenon was detected during most types of work.

It is interesting to note that controlled behavior studies going back to 1938 (8) showed that three things happened to people in a stimulus-response situation when the stimulus became both long and erratic. First, they slowed down in their work, second, they became emotionally upset; third, they made more mistakes.

If we were to tolerate a system response time as long as two seconds in our laboratory and assume Boies' phenomenon occurs, it would cost us a minimum of 36 million seconds per month of lost human time. That is 10,000 man-hours, or 60 people lost full time for the month. It is our objective to keep 90 percent of our interactions to a response time of .5 seconds or less. Subsecond response time is an important human requirement.

Figure 5 shows the distribution of user response time (URT). Others have called this *think* time. The mode of this distribution is 2 seconds. The median is 8.5 seconds and the mean is 12 to 15 seconds, depending on when one terminates the data points. If all end points, such as the two hour lunch break, or the time when users forgot to log off, are included then nearly any average is conceivable. In the extreme case what is really being measured is how long the system stayed up. But, by truncating after two minutes we find an average URT of 12 to 15 seconds, and we have observed that average for the past several years. The high frequency of points in the two second range may reflect the fact that people scan far more information than they read, and read more than they write.

Figure 6 highlights the impact of scheduling on the man-machine interface. It shows the impact of the Resource Management PRPQ for VM/370 in a benchmark environment as it would affect users. The columns titled RM PRPQ show the results of running with the PRPQ in an heavily loaded environment. The

columns titled RELEASE 3 show that same heavily loaded environment without the PRPQ. Notice that even in this heavily loaded environment 50 percent more interactions can be handled by the system with the PRPQ. And the response time at the 90th percentile is 20 times better. Notice also, that the base system handles more interactions with 60 people than with 80. By allowing those extra 20 users onto the system in that overloaded environment we observe that we have effectively lost the 20 people and slowed the other 60 down. In other words, without a knowledge of the expansion factor the number of simultaneous users being served is a poor measure of the load on the system.

Figure 7 shows computer cost on the vertical axis and the expansion factor on the horizontal axis. If the machine is underused, then most of its cost may be lost dollars. As the demand for computing services grow, the expansion factor also grows. At some range of expansion factors, the system is being optimally used in the sense that as the load builds beyond that to higher expansion factors a significant portion of the hardware usage goes into unproductive work.

Figure 8 shows the lost hardware dollars per hour due to underload and overload on one of the VM/CMS systems at Yorktown. The V shaped curve is an envelope containing the actual operating points. The expansion factor used here is an actual elapsed time expansion factor. It is the actual time to do some typical unit of computer-limited work divided by the minimum time to do that work in a stand-alone environment.

Figure 9 shows the cost of the lost end user time per hour on the vertical axis. The horizontal axis shows the expansion factor. The expansion factor grows together with the number of simultaneous users, and the number of people being slowed down by the delivery of poorer service to each also grows.

Figure 10 shows the combination of the hardware operating points and the end user costs as a combined loss. Note that there is indeed an optimal operating range, having expansion factors in the range of 4 to 6. This

optimal operating range is dependent on both the operating system and the machine speed. The cost of overload is far greater than the cost of underload when the end user's time is considered.

We use VM Monitor to monitor the daily use of our VM/CMS systems in Yorktown, and we attempt to maintain the operating point very close to an expansion factor of 5. Note that this definition of an expansion factor, using elapsed time to stand-alone time, differs by about a factor of three from that reported via the IND command in VM/370. The IND command normally reports a number that considers all users in a single server environment.

W. H. Tetzlaff has developed a very sharp set of analysis programs to analyze the service each user receives each day. These programs are run each night, and give us a clear indication of the quality of service, what caused problems, and what the relative pressure is on each system resource. His programs calculate the expansion factor for each hour of the day. These programs make the business of tuning, capacity planning, and service analysis a fairly straightforward task. They are described in his paper 'State sampling of Interactive computer users'.

W. J. Doherty [9] has studied these issues extensively and has defined the *performance* of an entity (a function, a system, etc.) as the degree to which the behavior that entity meets the observer's expectations. More precisely, performance is the difference between the observed behavior of an entity and the observer's expectations of behavior. We have found that if we can give the computer user some notion of the current state of the system he is planning to use, his expectations are modified accordingly and he can make sensible decisions in planning his work. For example, he may elect not to log on to the loaded system but to do his computing later when the expansion factor is lower. This decision on his part avoids making unsatisfactory service even worse. In TSS we introduced a numerical measure called the THI (based on the U. S. Weather Bureau's Temperature-Humidity Index of human comfort) which was an averaged value of the time needed to perform a standard task. The THI was available to the user before he logged-on, and he interpreted that

number in terms of what he was planning to do, e.g., editing would get reasonable response even with a relatively high THI, but a PL/I compilation and execution might be unacceptably slow under the same conditions.

It was reasonable to carry to VM/370 our experience with the THI but it was also desirable to give VM users more information about the status of VM than the simple THI of TSS. Accordingly, P. H. Callaway [10] implemented a THI for VM. The user must logon to VM/370 to obtain this information, but VM logon is usually very rapid and consumes very little resource. The components of this THI are available to both the control program and a user program executing in virtual storage. Thus, certain maintenance tasks such as data base back-up or data migration can check for low system activity before proceeding, which tends to minimize their impact on the user community. We have also found that by giving the operator information about system overloads derived from this THI, he can ask dominating users to postpone or moderate their activities, and he has been able to detect program looping not apparent to the user. The Watson Research Center version of the VM/370 THI together with other information collecting facilities we had implemented were made part of VM/370 Release 2 as the VM/370 Measurement Facility. VM/370 THI is now known as the Load Indicator.

Data Management

In 1965, APL on the Yorktown Computing Center's 360/50 imposed rigid constraints on the size of the user's data base: 36K bytes of working space of which about 32K were available to the user, who constantly protested against those limits. At that time the only way to increase work space size in APL was to provide more main memory and/or reduce the number of workspaces in main memory simultaneously (thereby usually increasing response time). Until APL/CMS became available, APL work space limitation was an annoyance to many users. On TSS/360 the situation was radically different because to its users TSS appeared to be a one level store. To the limits of allocated disk space, the TSS user could create data sets and file them and

retrieve them by data set name. The system managed all direct access storage space for the user. We could not continue to add on-line disk storage to TSS, but we also wanted to avoid requiring the user to spend time moving data sets to and from private tape or disk volumes. We developed a data set migration facility for TSS which became operational in April 1969, and had the following characteristics:

(1) Data sets were marked with a date last referred to.

(2) Data sets not referred to for more than N days (the value of N determined to keep a safe amount of on-line space available) would be compressed and migrated from the on-line disk volumes to demountable disk volumes stored off-line.

(3) If the user referred to a migrated data set, the system informed him it was migrated and told him how to get it back.

(4) The user could list the names of all migrated data sets, voluntarily migrate a data set not needed for a while, and erase unneeded data sets.

TSS Data Migration met many of the user's data management requirements but had some limitations. There was no mechanism by which a user could specify that any migrated data set to which he referred be restored automatically without additional action. This was an annoyance even though the user was given a message containing the name of the migrated data set. If the user knew that he must restore a migrated data set, he still was required to initiate the restoration process and to wait until restoration was complete. Most users preferred that restoration proceed without locking the terminal so that other work could be done. We have learned that for future data migration systems, when restoration is completed the user should be informed.

With the introduction of CP/67 and later VM/370 at the Watson Research Computing Center, the interactive system user found that the burden of on-line space management was transferred back from the system to the user.

(1) Because he is given a fixed allocation of disk space, the user might have to stop in the middle of his work to decide which files to erase or move to tape in order to make room for new files being created.

(2) As his files grow, the user must try to persuade installation management to give him more space; on the other hand he is pressed to give up space he is not actually using.

(3) When the user is given more space, his previous allocation must be copied by the Computing Center into the new space. This is an overnight operation, and usually requires that many users' space allocations be copied even if they are not changed in size.

By late 1972, the Computing Center had more requests to add new users than we had available disk space. Since user disk space on VM is frequently only partially used, and since on-line disk space is only fractionally active at any point in time, rather than invest heavily in additional disk storage devices, management explored two strategies. First, and least acceptable to the user community, we imposed an external schedule which distributed the available system time among the users by mounting only certain subsets of users' disk files at certain times. This not only constrained each user to specific hours, but frequently made it impossible for him to access files of a colleague if those files were only available at a different time.

Second, we began a one man development effort to determine whether or not user files could be managed by the system so that only "active" files occupied on-line disk space. This design was very promising and by 1973, we decided to carry our experience from TSS to the VM minidisk concept in VM. The most convenient unit of space on VM is not the user's file (although we subsequently implemented file migration) but his minidisk, a specific number of contiguous cylinders of on-line disk space. VM Data Migration as implemented at the Watson Research Computing Center moves and compresses an entire inactive minidisk on to a demountable disk, thereby leaving an "empty slot" on-line into which an active minidisk of that size can be moved on demand. As in the case of TSS, the decision to migrate a minidisk is made on the basis of a specified number of days elapsed since the minidisk was last accessed. The user whose minidisk has been

migrated experiences a longer than normal delay during his first logon while the minidisk is located, decompressed and moved into an empty slot on-line. Subsequent logons will proceed normally unless the user again fails to logon within the specified number of days.

Since the minidisk migration task must be available at all times, we implemented an AUTOLOGON facility which automatically logs on specified tasks after a system restart. AUTOLOGON has had value far beyond its role in minidisk migration; as a result of demands for asynchronous processing mentioned earlier, we find there are many special tasks which carry out important services for our users, *e.g.*, the NETWORK task enabling users to send work between systems, and which required manual restart by the operator. Now they are automatically started at system load time. If the operator failed to restart these special tasks, the user found these important functions unavailable.

The file migration capability which we have added to VM/CMS enables the user to migrate a file by *file name* from his minidisk to a demountable disk. In this way he can save and retrieve CMS files.

A valuable consequence of VM migration is the back up capability it affords the installation. On third shift, copies are made of all minidisks accessed on the previous day, and are stored on a second set of on-line disks which are copied to tape. If a user inadvertently erases his files, within four hours the Computing Center is able to produce a copy of his minidisk which is no more than twenty-four hours old. The Computing Center has very rarely lost a user's data; nearly all data losses take place when the user erases or inadvertently writes over his own files. We believe that the Computing Center has a continuing responsibility to safeguard the user's data against both accidental and unauthorized access or destruction.

Another major benefit of the data migration facility is the reduction in cost for managing on-line data. Just before the introduction of these facilities for VM/CMS in 1973, there were about 350 tape mounts per day and 200 disk mounts per day across the whole installation. By early 1978, when the use of computing had grown by a factor of 8 times, we were processing 30 tape mounts per day and 5 disk mounts per day. Thus

an effective reduction of at least a factor of 80 times was achieved in the frequency of tape and disk mounting.

Distributed Function

A user's CMS vitrual machine in VM is well isolated from other CMS virtual machines in the sense that a user may cause his CMS machine to "crash", but does not affect others. Therefore it was our belief that the CMS machine would be an excellent vehicle for the development of special functions which could then be tailored for ease of use, ease of maintenance, and good performance. The strength of VM lies mainly in the strong isolation provided by virtual machines.

Because we could use CMS to isolate a particular function from all other functions, the user need learn only that function to use that function. Thus, complexity in the end user interface is lessened. As a function evolves over time, tradeoffs in efficiency and ease of use can evolve together without being clouded by other, non-related issues.

Because the function is isolated to a virtual machine, it becomes easy to add new versions either as copies or replacements. This greatly reduces the sensitivity of the whole system to changes. In addition, the VM/370 system automatically re-IPLs itself in most cases if it should fail. It then automatically re-IPLs all such special purpose virtual machines. This is normally done without operator intervention and typically takes about 30 seconds of elapsed time.

Because the users of interactive systems grow in what they use the system to do, we find that interactive growth results in a proportional requirement for asynchronous processing, i.e., the ability to do computing

which is separated in time from the user's interaction with the system. This had often been called batch processing in the past. By grouping special purpose functions in special machines, we can apply better controls to those functions. i This includes special scheduling controls, grouping of requests to reduce overhead of initialization, and tailoring of the function over time in accordance with the user's evolving needs. Also, by giving the user the ability to submit work to asynchronous virtual machines, we remove bottlenecks from the user's interface to the system and his data. It is important to note that these special purpose functions need not run on the same real machine as the user is currently using.

Thus, the virtual machine concept directly applies to distributed processing. The gains to be had for load distribution are real, but they are small in comparison to the gains in reduced complexity, ease of maintenance, and improved service to the end user.

Our experience with distributed function leads us to the conclusion that we have now seen three distinct stages of computing center activities. The first occurred in the early sixties, when the computing center provided computing services and served as a pool of applications programmers allocated to help scientists develop their programs to use the computer to solve problems that were machine limited. This was because computing was about 100 times more costly at that time, and so only a few applications were cost justifiable. It was also due to the lack of data management and primitive user interface then in existence.

The second stage was that of the system programmer when most computing centers grew rapidly in size and the systems did also. Highly skilled people were required and change management grew increasingly complex. The computing center personnel were devoted to make the systems more manageable. However, their time no longer went directly to the end user. He became a direct user of the systems.

The third stage is our current environment in which the systems personnel are primarily involved in some basic system changes and the management of tools for distributing the responsibility of change control to the

users who have the strongest need for the changes. In this way the end users themselves can better control their own environment. Our networking capability allows our users to communicate freely between one another and send new functions directly to each other in machine readable form. Our users send an average of the 3,000 files a day in and out of Yorktown computing systems. The typical file size is 50,000 characters, which is the equivalent of a 20 page paper. That is a measure of the communication among users on physically different real machines. In fact, the communication among users of the same physical machine, but different virtual machines is probably far greater.

Approved For Release 2001/07/12 : CIA-RDP84-00933R000500120001-5

Summary

Appropriate management actions can significantly enhance the effectiveness of an interactive system from the user's point of view. During more than ten years experience with interactive systems at the IBM Thomas J. Watson Research Center, we have made management decisions which extend the capability of these systems to work for the user by enabling him to make sensible decisions about how he will use his time, and by selectively adding to the systems facilities which enable him to spend less time compensating for system limitations and more of his time on his problem. Increased emphasis has been placed on using the computer as a tool to extend the users' memory as well as their reasoning power. Our computing systems are adapted by experienced users, via EXEC's, so that the effectiveness of man-machine communication is increased with time. This is of special benefit to the inexperienced user who then takes advantage of the experienced user's evolutionary growth. Issues of availability, distributed change management, data management, inline documentation, response time, expansion factors, distributed function, and effective user to user communication have been key to our success. VM/370 and CMS are the primary vehicles that our users have found to be effective for our rapidly evolving interactive environment.

Acknowledgements

The managers of VM/CMS at Yorktown Heights have contributed much to the IBM product in addition to effectively providing excellent service for the Yorktown users. These include W. M. Buco, A. N. Chandra, B. Lie, and N. J. Pass. W. H. Tetzlaff and P. H. Callaway have contributed much to the performance management of VM/CMS and their work is gratefully acknowledged. L. H. Wheeler has developed scheduling and resource management strategies that have sharply increased our ability to provide interactive computing service to a large number of people at Yorktown.

REFERENCES

[1] A.W. Luehrmann and J.M. Nevison, Computer Use under a Free-Access Policy, Science 184, 957-961 (1974)

[2] W.E. Daniels and R.W. Ryniker, EXEC 2, A Computer Language for Word Programming, IBM Research Report RC 6292, November 17, 1976

[3] W.J. Doherty, Human Factors: Impact on Interactive Computing, Proceedings of SHARE 50, vol. 2, 1244-1266, March 1978.

[4] Private communication between T. Johnston (SLAC) and W. J. Doherty, Feb. 1978.

[6] T. R. Bell,

[6] A. Guido and J. P. Considine NCC paper on TPVM

[7] S. J. Boies and J. D. Gould, User Performance in an Interactive Computer System, Proceedings of the Fifth Annual Conference on Information Sciences and Systems, Pg. 122, 1971

[8] R. S. Woodworth, Experimental Psychology, Henry Holt and Co., Inc., 1938

[9] W. J. Doherty, Measurement and Management of Interactive Computing, Proceedings of SHARE XLIV, vol. 3, 1587-1598, March 1975.

[10] P. H. Callaway, A Performance Measurement Approach for VM/370, IBM Research Report RC 4666, January 4, 1974.

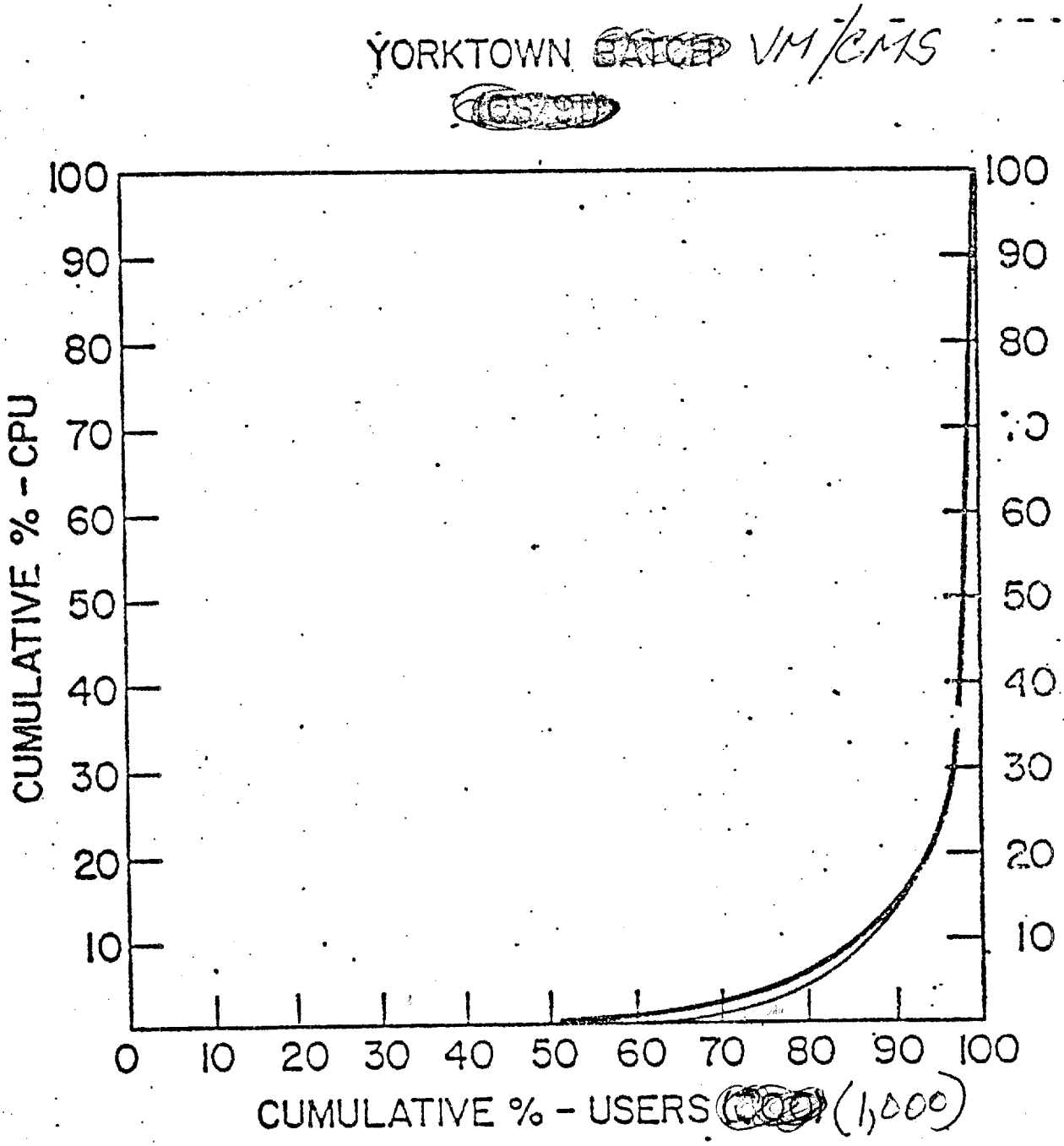
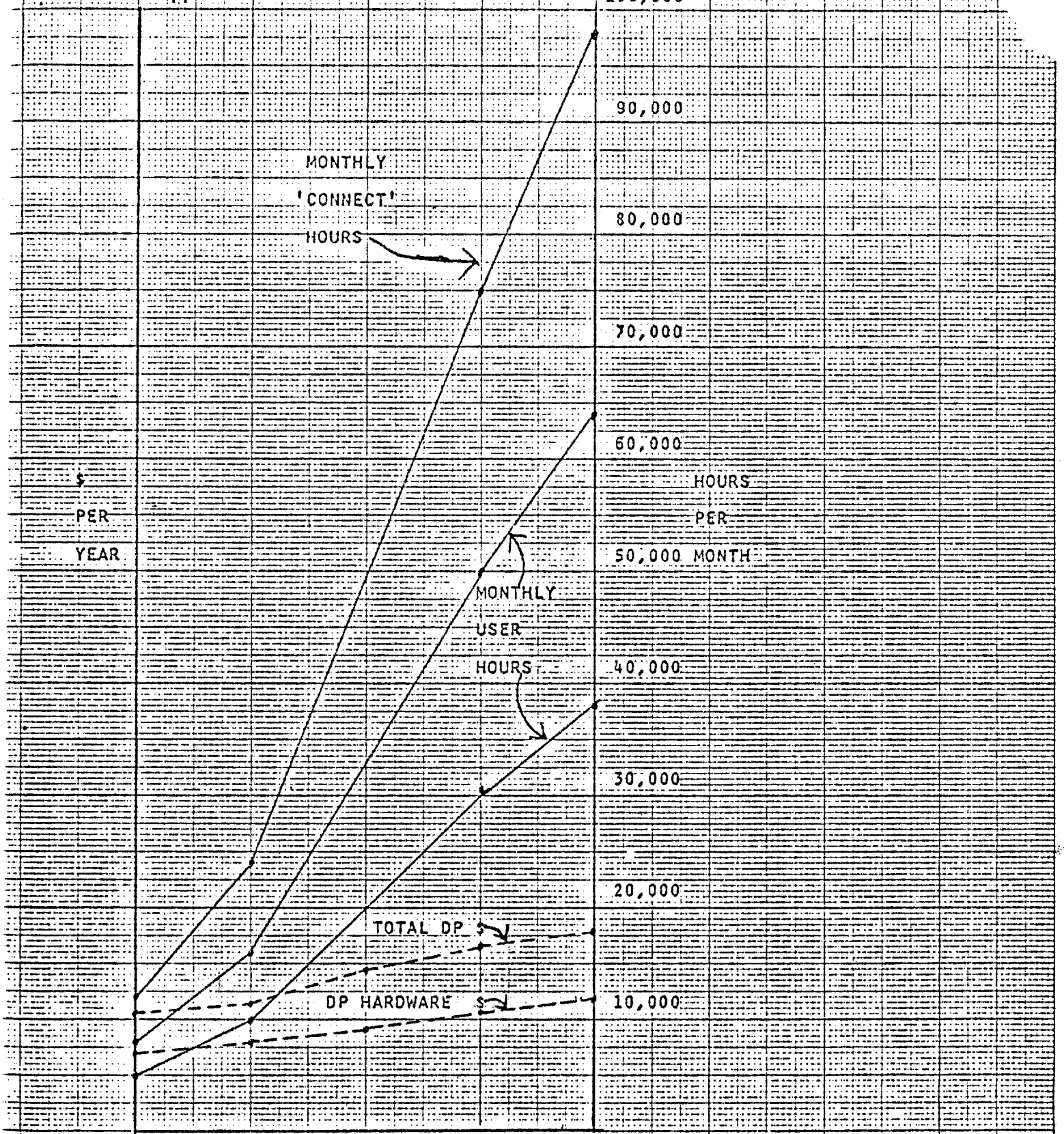


FIGURE 1



IBM T. J. WATSON COMPUTING CENTER

COMPUTING COSTS vs. USER COSTS

FIGURE 2

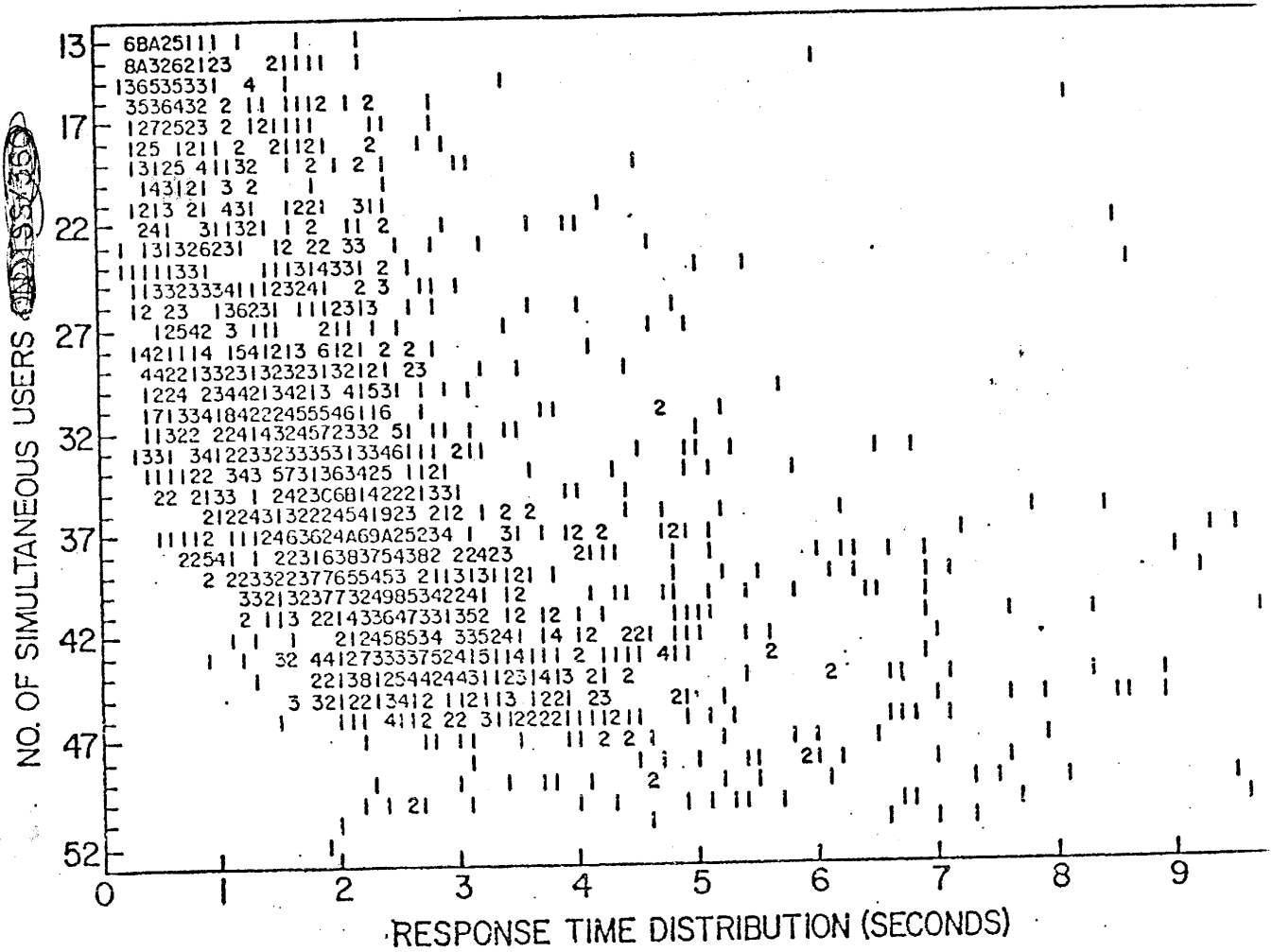


FIGURE 5

USER RESPONSE TIME IS DIRECTLY INFLUENCED

BY SYSTEM RESPONSE TIME (SRT)

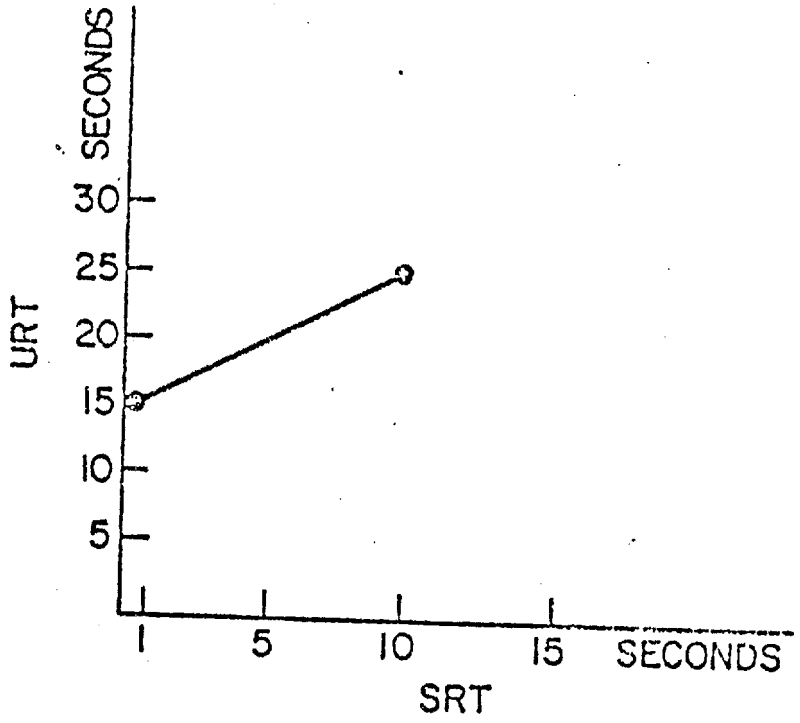


FIGURE 4

FREQUENCY OF USER RESPONSE TIME

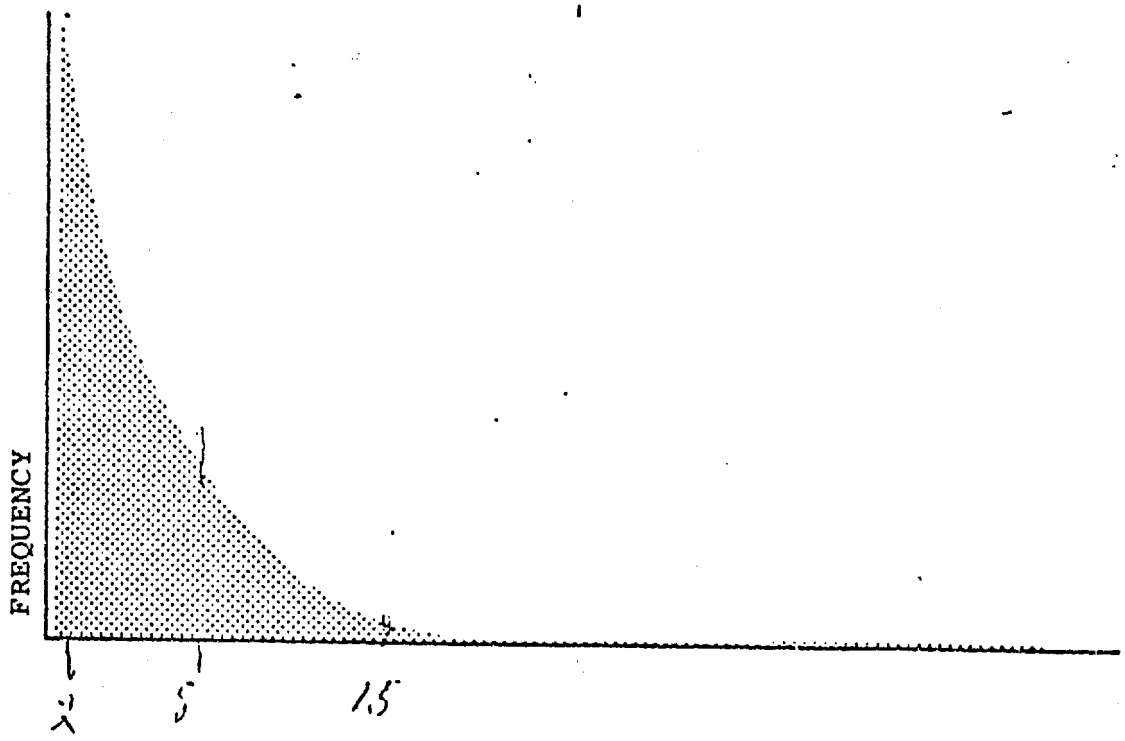


FIGURE 5

158 CAMBRIDGE BENCHMARKS (40 MIN.)

	WHEELER		RELEASE 3			
MEMORY	2 MEG	1 MEG	2 MEG		1 MEG	
RESPONSE	0.146	0.624	2.2	0.537	1.6	0.601
TRANSACTIONS	12,374	11,149	7,921	9,176	7,319	8,062
PROBLEM CPU	60.4%	50.3%	33.6%	37.1%	23.8%	43.5%
TOTAL CPU	93.5%	88.8%	66.6%	97.4%	51.2%	73.2%
DRUM I/O	95%	95%	65% →			
DISK I/O	5%	5%	35% →			
USERS	80	80	80	60	80	60

FIGURE 6

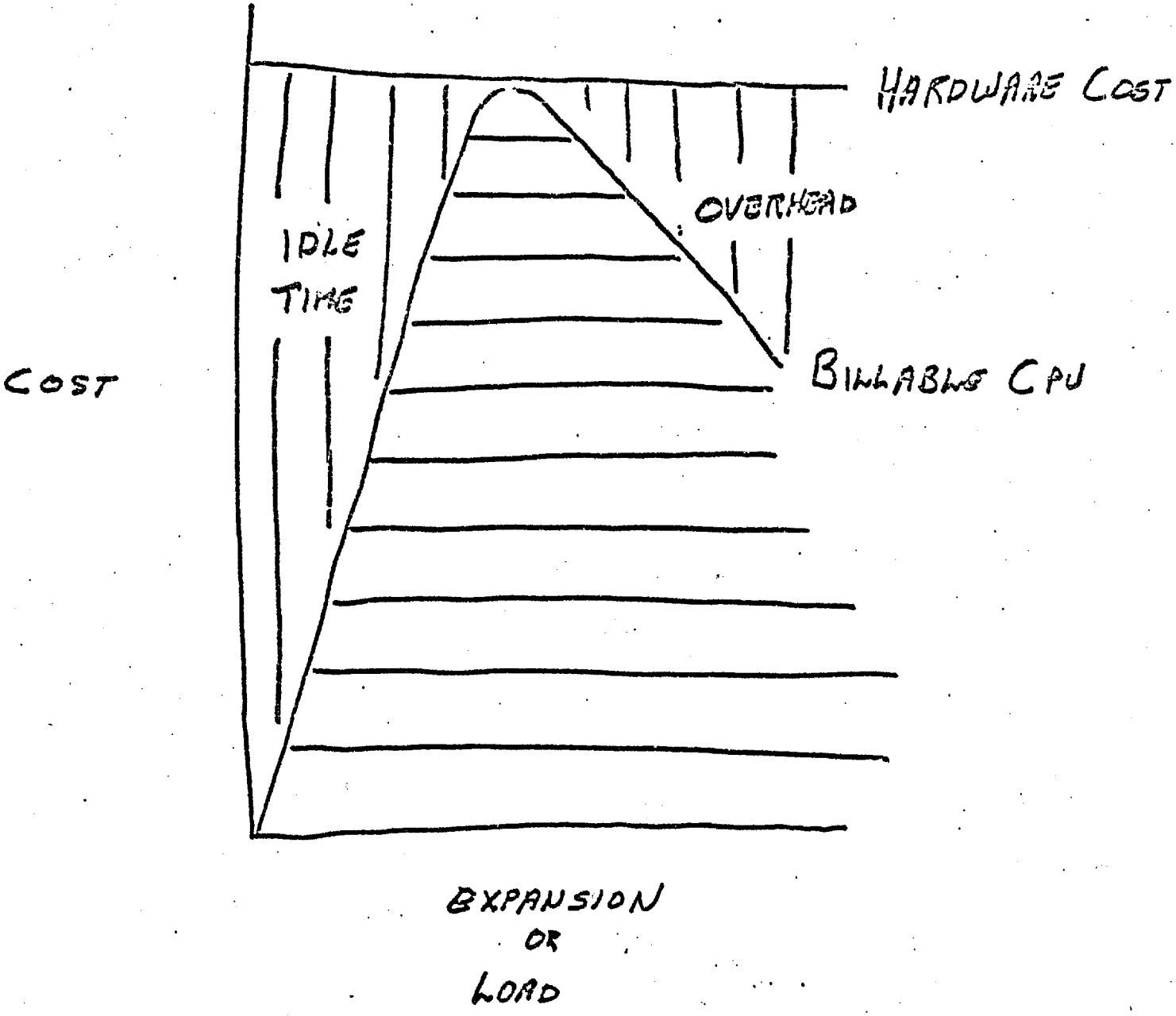


FIGURE 7

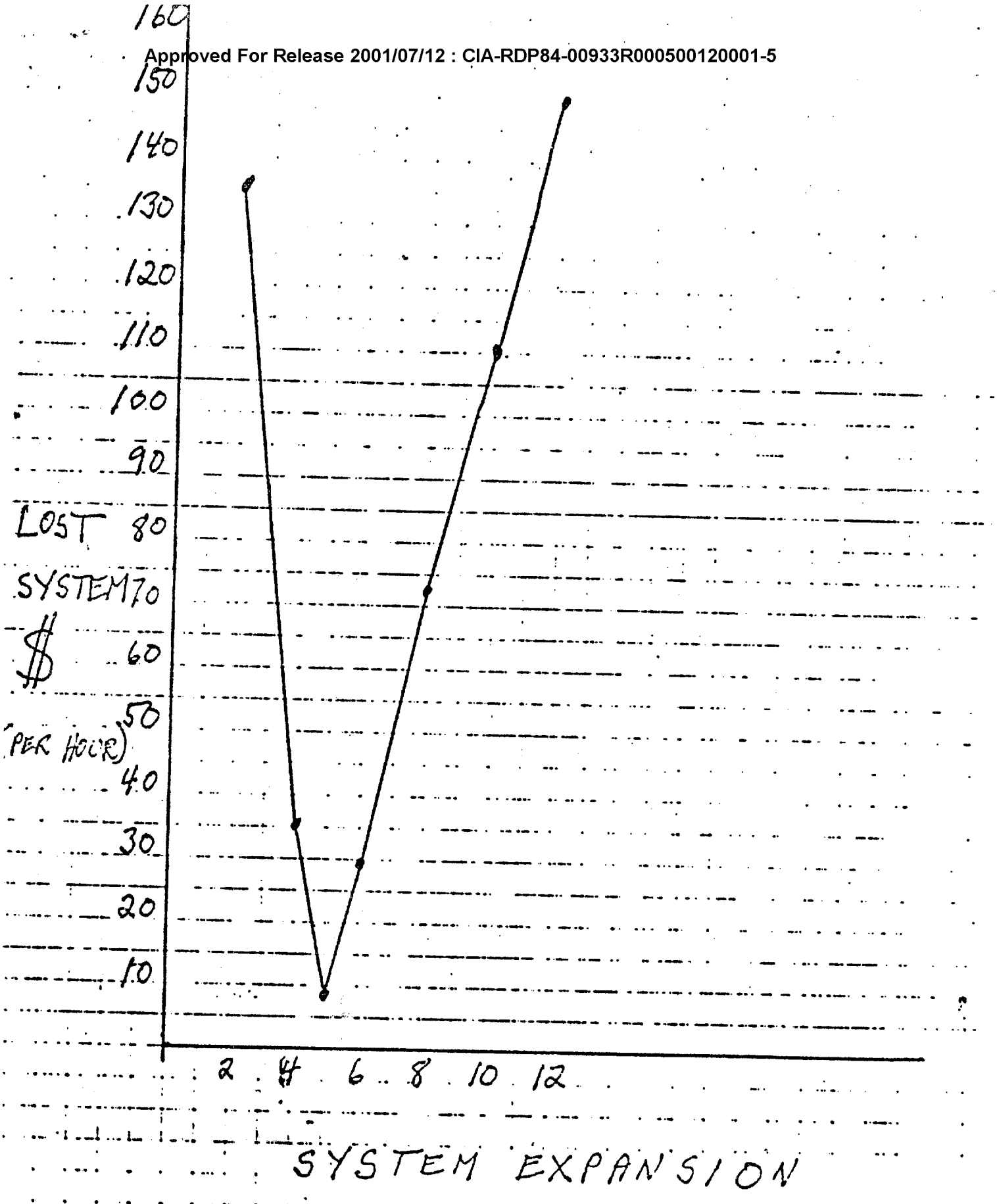


FIGURE 8

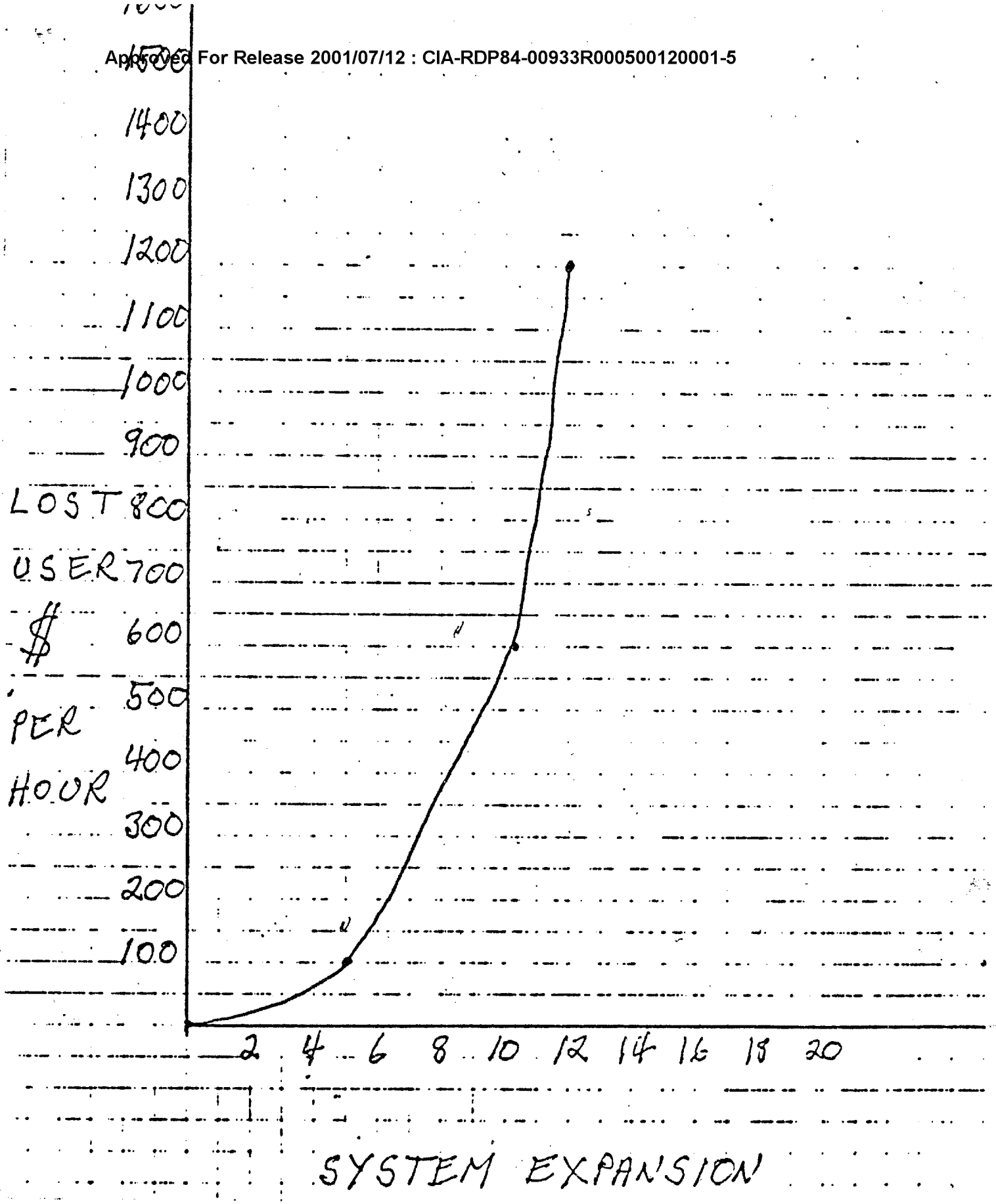


FIGURE ~~1~~ 9

Approved For Release 2001/07/12 : CIA-RDP84-00933R000500120001-5

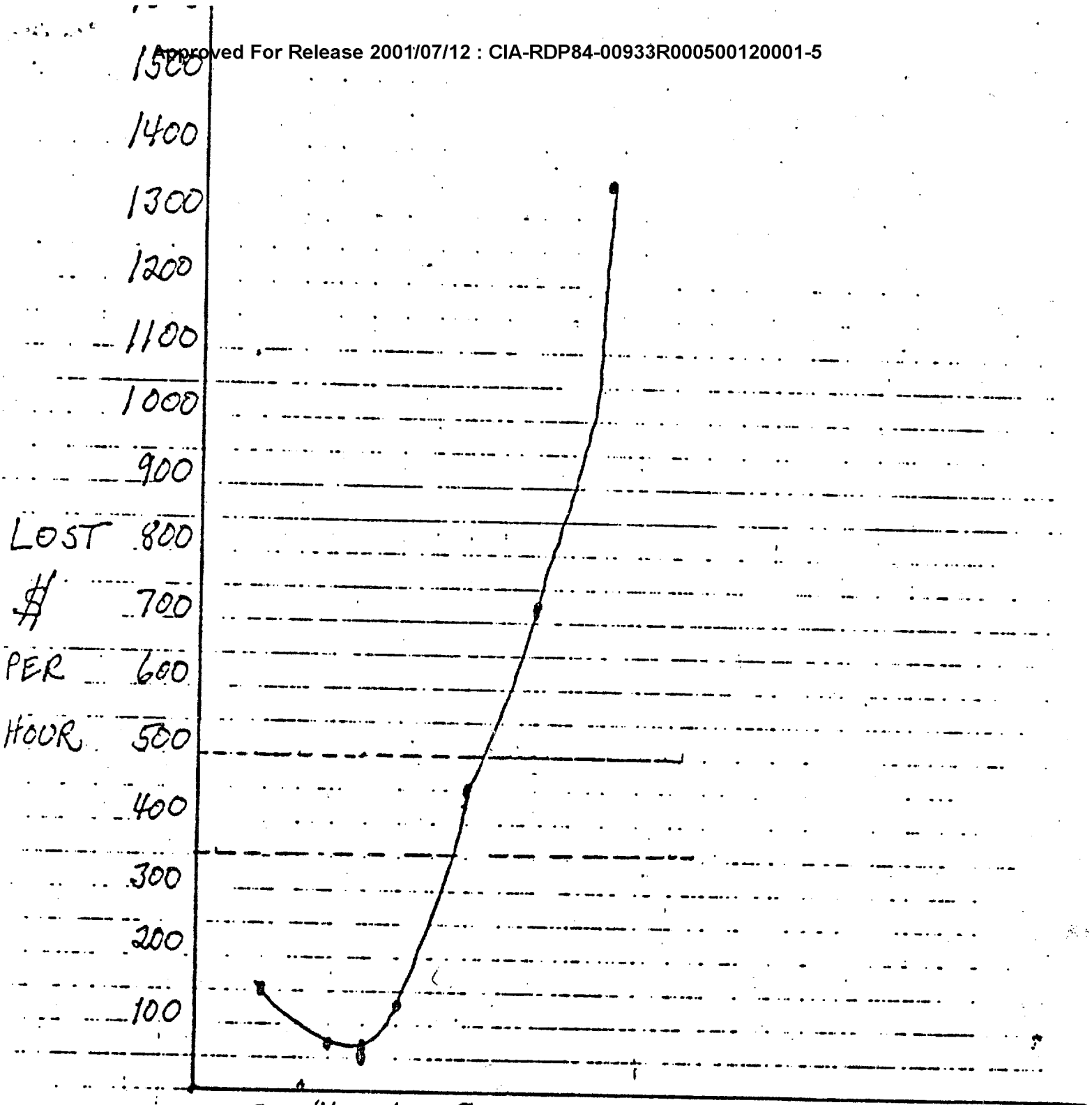


FIGURE 10

HUMAN FACTORS: IMPACT ON INTERACTIVE COMPUTING

W.J. Doherty, IBM Research, Box 218, Yorktown Hgts, N.Y., 10598, U.S.A.

This presentation will describe the human side of interactive computing as seen in the Computing Center at IBM's T. J. Watson Research Laboratory in Yorktown Heights, N. Y. This computing center has been a SHARE member (PK) since 1958.

Figure 1 describes the mission of this computing center. We find that complexity is the single biggest problem facing the end user today. Since 1968, we have looped on the last two items in our mission. We made the decision at that time to stand at the man-machine interface and look outward to study the impact on people of the computing function which has evolved over time. We learned of a number of technical and administrative barriers which got in people's way. We worked hard to remove those barriers.

Upon removal of these barriers, we can make the following observations:

The value of today's computing systems to their end users is in their DATA, PROGRAMS, and PROCEDURES. Libraries of thousands of programs exist in all major installations. The user interface is simplified by combining programs together and burying default parameter specifications inside PROCEDURES. PROCEDURES outnumber all programs written in conventional programming languages.

The issue of integrating the DATA RESOURCE into the enterprise, (building bridges to tie the different subsystems together) will probably be the dominant driving force in computing over the next ten years. (See Dick Nolan's paper 'Thoughts about the fifth stage'.)

Figure 3 shows the current picture of the computing configurations at Yorktown. Notice that all systems can communicate with each other and use the MSS as a common back end. Terminal growth has been rapid, especially since the advent of the IBM 3270. Laboratory automation, program development, and text processing have all grown rapidly as well. VM/CMS is the interactive interface which is the most popular in our laboratory.

At Yorktown, the computing systems have moved from being specific tools to solve specific problems to a general resource which extends people's memory and reasoning power across the entire spectrum of tasks they do. This is a NEW DIMENSION of computing use. It requires the removal of the barriers we encountered to achieve the levels of productivity we have today.

Figure 3A shows the growth in end user time relative to computing costs from 1973 through 1977 at Yorktown. The uppermost line shows the growth by a factor of 8 in monthly 'connect' hours. This includes dormant end user time, active end user time and special function virtual machine time. The next two lines down show the growth in end user time at the terminal. The uppermost line shows a growth from 8,000 to 64,000 hours per month. It is a reasonable upper limit on the amount of time that the interactive systems were influencing our people in their work. The lower line, showing a growth from 5,000 to 38,000 hours per month, is an estimated lower bound on the amount of time that our interactive systems had a direct influence on people's work. The next two lines down show the total computing cost relative to the end user cost, and the hardware cost relative to the end user cost.

Thus, small errors in DP equipment planning can result in sharp losses of end user time due to overload.

Figure 4 shows how complete the growth of interactive computing has been in penetrating the different departments we have in Yorktown. It shows the number of people in each department who spent many hours of their time working interactively in Nov. 1973, and again in November, 1976. For example, the first column shows that department 41X, our administration department, had three people in 1973 who spent from 20 to 40 hours per month of their time working interactively. It also shows that there were 38 people in that department who spent from 20 to 40 hours per month of their time working interactively in November, 1976.

Yorktown processed at least 7,000,000 and at most 11,500,000 interactions per month in November, 1976. By November, 1977 that range had grown to at least 9,000,000 and at most, 15,000,000 interactions per month.

Figure 5 shows the number of people in the Laboratory who spent different amounts of their time working interactively in the month of November 1976. Today, over 25 percent of our Laboratory is working interactively at each instant throughout the day. Over 75 percent of our Laboratory works interactively in each week.

Figure 6 shows the difference in intensity between an interactive and a batch environment. Because interactive computing is so much more intimate in its impact on the end user, we have learned that functional availability must be targeted for 24 hours per day and 7 days per week. If information is not available from a computing system when it is needed, then people do not keep all their data online. This also implies that maintenance must be done time-shared.

In 1971, we saw that the number of commands per person doubled in just 5 months, when counted after procedure expansion.

At the SLAC (Stanford Linear Accelerator Center) installation today, using the Wylbur interface, there are an average of 25 commands executed for each command typed at the terminal.

People learn how to use a computer to perform some function for them. They then ask the computer to remember how to do that. As people grow in experience with such a system, they combine functions to raise the information processing bandwidth between themselves and the computer. It is hard to forecast how long this growth will continue. It seems evident that the quality of the command procedure facility, which is the software glue to help this growth, is critical to user growth.

Figure 8 shows the number of different file types on one of the VM/CMS systems at Yorktown. Of the 800 people who use this system, this chart shows the files belonging to 164 of them. In some sense, that's about 20 percent of the users of this system at Yorktown. Notice that 162 of the 164 users had a total of 6074 EXEC files. If we look at all other forms of programming language we get a total of 3000 programs in source form. EXECs are used to raise the level of the man-machine interface by combining programs together. They reduce complexity by allowing many parameter values to be decided automatically. They reduce errors by eliminating the need for people to reenter the same commands and parameters time and again.

As people learn how to do more and more things with computers, EXECs help that growth. EXECs are probably the single most important facility in the system to aid user growth.

Figure 9 shows the importance of the software component known as the editor. Editors are the single system component where the computer user spends the most time. Estimates are that 50 to 80 percent of people's time is spent editing text, data, or programs.

Editing programs have historically not received much attention as formal computer software products. However, we find ourselves with a great proliferation of EDITORS today. Carol Thompson's USER'S GUIDE TO THE RESEARCH CONTEXT EDITOR--REDIT, IBM Research report No. RA 28A, contains an excellent history of the evolution of one such editor program.

It seems that the human factors associated with editor programs are therefore of great importance.

Figure 10 shows a picture of the response time to a single small program which ran every 20 seconds in the spring of 1971. Each 20 seconds, it woke up, looked around at the other users on the system, and recorded the response time that it received from the system. This is really a 3 dimensional graph. Each plot position is a number indicating the number of times that operating condition was found to exist.

Notice that as the number of simultaneous users increases, the response time steadily, and sharply degrades. Thus, if we simply try to maximize the number of simultaneous users, we find that we are maximizing the number of people to whom we are presenting a sharply degrading picture of service.

Remembering how rapidly people grow in their use of programs, it is clear that a given number of people will seldom ask for the same work to be done.

If we recall that user growth results in a sharply increasing number of programs invoked per person, and that service degrades beyond some load threshold, we should realize that the number of simultaneous system users is a misleading measure. It is indeed a measure of potential liability. It is not clear what else it measures.

We find that most genuine performance improvements lead to a direct reduction in the number of simultaneous system users. This is accompanied by an increase in the number of interactions processed. There is also an increase in the number of people served over a longer time such as a day or week. The simultaneous congestion has decreased.

Figure 12 shows the profound influence that system response time degradation can have on user behavior. If we break an interaction into two parts, the system response time (SRT) where the system is processing a request for the user, and a user response time (URT) where a user is inputting his or her next request to the system; we find that each second of system response degradation leads to a similar degradation added onto the user's time for the following request.

This phenomenon seems to come from people's attention span. The traditional model of a person thinking after each system response appears to be inaccurate. Instead, people seem to have a whole series of actions in mind, contained in a short term buffer in their heads. In fact,

they are continually thinking. Disruptions to their thought processes are generated by delays in SRT. This results in their short term memory buffer being reset.

It was first discovered by S.J. Boies at Yorktown in 1971 while doing normative studies of interactive computing. We had recorded every interaction on TSS for 3 years. It included a wide range of human tasks. The phenomenon was consistently repeatable across most types of work.

In 1936, controlled behavioral studies showed that three things happened to people in a stimulus response situation when the stimulus became both long and erratic.

First, they slowed down in their work.
Second, they became emotionally upset.
Third, they made more mistakes.

If we were to tolerate a system response time as long as two seconds in our Laboratory today, it would cost us a minimum of 36 million seconds per month of lost people time. That is 10,000 man-hours, or 60 people lost full time for the month. It is our objective to keep 90 percent of our interactions to a response time of .5 seconds or less. Subsecond response time is an important human requirement.

Figure 13 shows the distribution of user response time (URT). Others have called this THINK time. The mode of this distribution is 2 seconds. The median is 8.5 seconds and the mean is 12 to 15 seconds, depending on when you terminate the data points. If all end points, such as the two hour lunch break, or the time when people forgot to log off, are included then any average is conceivable. What is really being measured then however is how long the system stayed up. By truncating after two minutes we find an average URT of 12 to 15 seconds. This distribution has held up for twelve years now. The high frequency of points in the two second range reflect the fact that people scan far more information than they read, and read more than they write. There is a real need for computing system response times on the order of one or two tenths of a second.

Figures 14 and 15 show the load on a S/360 Model 67 from running a few programs. They are a conservative two dimensional picture of load. CPU time is on the horizontal axis and main storage requirements in 4096 byte pages are on the vertical axis. The shaded area is supervisor state time, and the clear area is problem state time. The area under each rectangle is a conservative measure of load. It is conservative because the I/O dimension and the paging overhead dimension is missing. The difference in area varies by five orders of magnitude from the smallest to the largest. Almost all the variation in load comes from the amount of data being processed and not from the kind of processing.

A classic problem in our industry has been the problem of benchmarking. This has arisen as a problem by the confusion between load and function measures. For example, one installation said that they had characterized their load. They had a beautiful functional distribution of their work, but had only accounted for 2 percent of their load.

Load, that is, demand on the CPU, I/O, and main storage, arises mainly from the amount of data being processed rather than from the nature of the processing. Variations in load on the order of five orders of magnitude are regularly seen for editing as well as compiling.

Measures of function are useful for productivity studies. They are of very little use in capacity planning.

Figure 16 shows service measures for two classes of work as seen on a S/370 Model 158 in one afternoon in July 1974 at Yorktown.

The horizontal axis shows the level of multiprogramming. It is the sum of the ELIGIBLE and DISPATCHABLE lists in VM. It is the parallelism in the input stream of requests. The vertical axis shows two different classes of service measures, depending on the nature of the work being done. For those interactions which could be completed within 200 ms. of 158 CPU time, the service measure is RESPONSE TIME, and it ranges from 0 to 2 seconds. For those interactions which were bigger, we use the EXPANSION FACTOR as the service measure. This is a multiplier that measures how much longer the work takes due to the sharing of the machine than it would in a stand-alone environment. The numbers across the top are the number of interactions in each class of work at each level of multiprogramming.

Notice that most of the interactions took place at a level of multiprogramming between 15 and 21. There is more simultaneity in the input stream than in the hardware.

Notice also that 97 to 98 percent of the interactions fell in the light load class of work. That is 97 to 98 percent of the people time, the most expensive commodity. The remaining two to three percent of the interactions accounted for 80 percent of the total demand on the CPU, I/O, and main storage.

By selectively detecting and scheduling the few large tasks, we can assure the remaining 98 percent of the interactions will have excellent response.

This has been done in the Resource Management PRPQ in VM/CMS.

Figures 17 and 18 show the distribution of CPU time over the user population at Dartmouth and in a pure batch environment at Yorktown. The horizontal axis shows the cumulative percent of the people using each system. The vertical axis shows the cumulative percent of CPU consumed by those users over a month. Notice that 95 percent of the users use a total of 25 percent of the resource. If we further examine the top 5 percent of the users we find that 5 percent of the programs and data they run account for 85 percent of their load. Again, a very small percent of the interactions account for the overwhelming majority of the load.

By putting a small effort into tuning these few users' large programs, we find large service gains can be achieved in short time periods. This seldom results in less overall load since you are taking a machine limited activity and making it less so. Most people simply run their programs more then. However, their service is improved sharply, and the granularity of the load they impose on others is smaller thus giving others better service as well.

It turns out that I/O has a similar distribution while main storage is slightly flatter.

Again, scheduling can be, and is effective in taking advantage of this load distribution.

I have seen many different systems in many different environments. All of them, except ACP, have the same load distribution. This is a reliable and repeatable load curve distribution.

Figure 19 highlights the impact of scheduling on the man-machine interface. It shows the impact of the Resource Management PRPQ for VM/370 in a benchmark environment as it would apply to people. The columns titled WHEELER show the results of running with the

PRPQ in an overloaded environment. The columns titled RELEASE 3 show that same overloaded environment without the PRPQ. Notice that 50 percent more interactions can be handled by the PRPQ in this overloaded environment. And the response time at the 90th percentile is 20 times better. Notice also, that the base system handles more interactions with 60 people than with 80. By allowing those extra 20 users onto the system in that overloaded environment we observe that we have effectively lost the 20 people and slowed the other 60 down.

Since people currently cost more than computers do, it seems that such measures should be incorporated into evaluations of systems.

Figure 19a shows computer cost on the vertical axis and the service measure called the expansion factor on the horizontal axis. If the machine is underused, then most of its cost may be lost dollars. As the demand for computing services grow, the expansion factor also grows. At some point, the system is being optimally used. As the load builds beyond that point, much of the hardware usage goes into unproductive work.

The next foil shows the lost hardware dollars per hour due to underload and overload on one of the VM/CMS systems at Yorktown. The V shaped curve is an envelope containing the actual operating points. The expansion factor used here is an actual elapsed time expansion factor. It is the actual time to do some typical unit of computer-limited work divided by the minimum time to do that work in a stand-alone environment.

Foil 19c shows the cost of the lost end user time per hour on the vertical axis. The horizontal axis shows the expansion factor. As the expansion factor grows the number of simultaneous users is also growing, and the number of simultaneous people being simultaneously slowed down by a slowly decreasing service to each also grows.

Figure 19d shows the combination of the hardware operating points and the end user costs as a combined loss. Note that there is indeed an optimal operating point, having expansion factors in the range of 4 to 6. The cost of overload is far greater than the cost of underload when the end user's time is considered.

We use measures like these to monitor the daily use of our VM/CMS systems in Yorktown on our two 168s. We find that we can normally maintain the operating point very close to an expansion factor of 5 for most days.

Figures 20 and 21 show the five layers of storage in today's systems and the problems addressed by managing each boundary in a better way.

By observation we have learned that data lives usefully in the cache for about 10 ms., in main storage for about 10 secs., on drum or fixed head file for about 10 minutes, on disk for about 10 days, and on archival storage, such as MSS, for many years.

The cache-main storage boundary has to do with CPU speed. Differences ranging from 2 to1 up to 5 to 1 are achieved by proper cache management. This is because the locality of reference of most programs is such that a very small part of the storage receives 95 percent or more of the action.

The main storage - drum or fixed head file boundary has to do mainly with system response time. The issues here are the working set characteristics of programs to be run, the page replacement algorithms of the operating system, the parachor characteristics of the combination of these two, and the management of the paging device channel program. Differences greater than 10 to 1 in interactive response time can be achieved by proper combinations of these four functions.

The drum - disk boundary has to do with the management of virtual storage. If there are 200 simultaneous users of an operating system, and each one has 2 megabytes of virtual storage, that is 600 megabytes of virtual storage. The 2305-2 fixed head files (drums) hold 11 megabytes of virtual storage on each drum. It clearly doesn't all fit on even 4 drums. However, the same locality of reference properties apply here as in all other levels of this hierarchy, only the time scale is on the order of ten minutes. Once again differences ranging from 2 to 1 up to 5 to 1 are achievable by a good drum-disk migration scheme.

Finally, the boundary between online disk storage and online archival storage is the key to user growth and user productivity savings. Again the locality of reference properties apply and we find that 95 percent of online disk storage is never referenced for periods of time greater than a day. Furthermore, it has been found that users who fill their allocated online disk space spend a fair amount of time being data space managers from then on. The gating factor is not the amount of disk space they have. It is the users themselves. They inevitably free up just enough disk space to do what they have to each day. This is counter productive on their part. However, the most important problem of all is the issue of user growth. By automating the disk - archive boundary an open ended secondary store effect is created. This opens up many new kinds of applications and a whole new way of using computers. At Yorktown the computing systems are treated as open ended extensions of peoples' memory and reasoning power. This applies across the entire spectrum of tasks they do. It includes a wide spread of people.

The previous Figures provide insight to the barriers and problems we have found and corrected. Motivated by what we have learned in terms of placing people productivity ahead of machine productivity we have realized that the need for asynchronous processing becomes very much stronger. If people are to grow in the use of computers to extend their memory and reasoning power, then obstacles in the path of that growth must be removed.

At Yorktown we use 5100s and the IBM 7406 for many Lab Automation problems. We built TPVM to interface System 7s for bigger Lab Automation problems. Computer Networking is used extensively to interconnect many different computing systems within IBM. In this way we can sharply reduce most conversion problems since we can nearly always find a site to run old programs. The message of COMMUNICATE DON'T CONVERT is powerful.

We have developed many special function virtual machines to address those functions which we found to be very common. In this way the special function evolves in an environment which is uncontaminated by other non related function. This sharply reduces complexity while simultaneously allowing the user to do other work at his or her terminal while the asynchronous processing of the special function goes on.

Some examples of this special purpose function are listed on Figure 22.

The rate at which our users increase their ability to work in this environment is most impressive. They first show improved quality of what they do. Second they show improved quantity. In general the quantity shows an increase in the neighborhood of a factor of 10 in Lab automation, document production, and program development.

Figures 23, 24, and 25 show a picture of the Research Device Coupler and several ways in which it can be interfaced to the systems. The RDC, which acts with, or as, a terminal, brings digital and analog input and output facilities to interactive terminal-oriented systems or to stand-alone systems such as the IBM 5100 Portable Computer. It is designed to provide those experiments with modest requirements access to high level language facilities, such as APL, FORTRAN, BASIC, PL/1, etc., for direct experiment control. This accounts for 70 to 80 percent of lab experiments. It eliminates the complexities of learning other operating systems.

Figure 26 shows some of the distributed function which we have in place today.

The lab scientist, using Labs/7 (EDX) as the operating system in his System 7 can control multiple experiments. When the need arises to use a more powerful computing system either for processing or for accessing a data base, a simple mechanism called TPVM provides remote intelligent sub-systems with the ability to access and utilize the power of the IBM VM/370 environment. The scientist can directly access and edit data in VM/CMS from his terminal on the System 7. He can also initiate asynchronous processing of data either totally independent from what he is doing, or in a way whereby he can monitor the progress of the number crunching while doing other things. The speed of the interfaces are such that we have seen response times ranging from 7 to 40 ms. for the user at the System 7 terminal.

As the need arose to interconnect the different computing systems with different operating systems within IBM, the Subsystems Unified Network (SUN) was developed. This has recently become a series of IBM products, known collectively as NJI and NJE. We now have the ability for users of more than 190 internal IBM systems to communicate data to each other, run jobs remotely, and disseminate the results to as many diverse systems and terminals as they need to.

By having locally attached display devices as terminals, our users can normally achieve sub-second response time for almost all that they do. At the same time they can communicate results, interchange technical documents, and freely communicate with colleagues all over the world. This communication takes place asynchronously from whatever they are doing at the terminal. A special function virtual machine manages the networking in our VM/CMS systems. Currently, VM/CMS systems constitute just over 2/3 of the systems in our SUN network. At Yorktown, about 500 people use the network to send and receive about 17,000 files amongst themselves and their colleagues on other systems each week.

The message COMMUNICATE DATA, NOT PEOPLE is powerful for interactive computing where the sub-second response time requirements are so strong.

Figures 27 and 28 show the early growth of the SUN network.

We have seen in earlier studies that editing accounts for 50 to 80 percent of what people do at terminals. One of the uses for editing is text processing. A special purpose virtual machine called TTF handles our text processing requirements at Yorktown. It provides the user with many useful facilities for composing documents. SCRIPT/370 is the primary formatting language used by our community. We have a compiler called STC (SCRIPT to TERMTEXT converter) which gives the user access to the more powerful TERMTEXT facilities, including over 400 separate fonts at the Yorktown computing center. Figures 29 and 30 show the sharp growth of the TTF service machine at Yorktown over the past year.

In summary, we have realized that the user at the terminal needs distributed function. In some cases this means special purpose virtual machines. In others it means interconnected computing systems. While in still others it means independently operating computing systems which may only be interconnected occasionally.

We have learned that users steadily and rapidly increase the number of diverse functions they perform and that the requirements for better asynchronous processing grow proportionally to the interactive growth.

For the past ten years we have studied the way our people used interactive systems and observed the barriers they encountered. They have been described in this presentation. The resulting working environment we have created for our Yorktown Research laboratory is highly productive and exciting.

Computer availability must be targeted for 24 hours per day and 7 days per week. If information is not available from a computing system when it is needed, then people do not keep all their data online. This implies that maintenance will have to be done time-shared.

Computer response time must usually be sub-second. This directly impacts people performance. Response time delays can lead to people delays, irritation, and more mistakes by people. Sub-second computer response time also opens up broad new areas of applications.

Data space management barriers, especially across main-drum, drum-disk, and disk-archive must be minimized. They are major barriers to economies of service and limits to user growth.

Complexity is the major problem facing the user today. By the use of EXECs we can sharply reduce complexity while increasing function. Also, by using virtual machines for special purpose function, we minimize contamination of the function, thus reducing complexity.

The VM/CMS systems at Yorktown are the major interactive interface to all forms of computing for the Yorktown user community.

Today, over 25 percent of our laboratory is working interactively at any time throughout the day. Over 75 percent of our laboratory works interactively each week. There has been a factor of 8 growth in the time our people spend working interactively in the past four years alone. Over 98,000 hours of 'connect' time is measured in a month today. The users are present, and subject to system influence at least 38,000 hours per month, and at most 64,000 hours per month. Most of the remainder is for the special function virtual machines.

MISSION

PROVIDE ADVANCED COMPUTER SERVICES

ENCOURAGE INNOVATIVE USES OF COMPUTERS

MAKE COMPUTERS EASY TO USE

EDUCATE USERS:

WHAT IS AVAILABLE

HOW IT CAN BE USED

UNDERTAKE SELECTED DEVELOPMENT PROJECTS

MEASURE, ANALYZE, AND EVALUATE COMPUTER SERVICES IN ORDER TO IMPROVE AND EXTEND THEM

FIGURE 1

ENVIRONMENT

INTERACTIVE USAGE GROWS RAPIDLY

MULTIPLE SERVICES AND LANGUAGES ARE ROUTINELY REQUIRED

REQUIRED FUNCTIONS MAY RESIDE IN DIFFERENT SYSTEMS

FIGURE 2

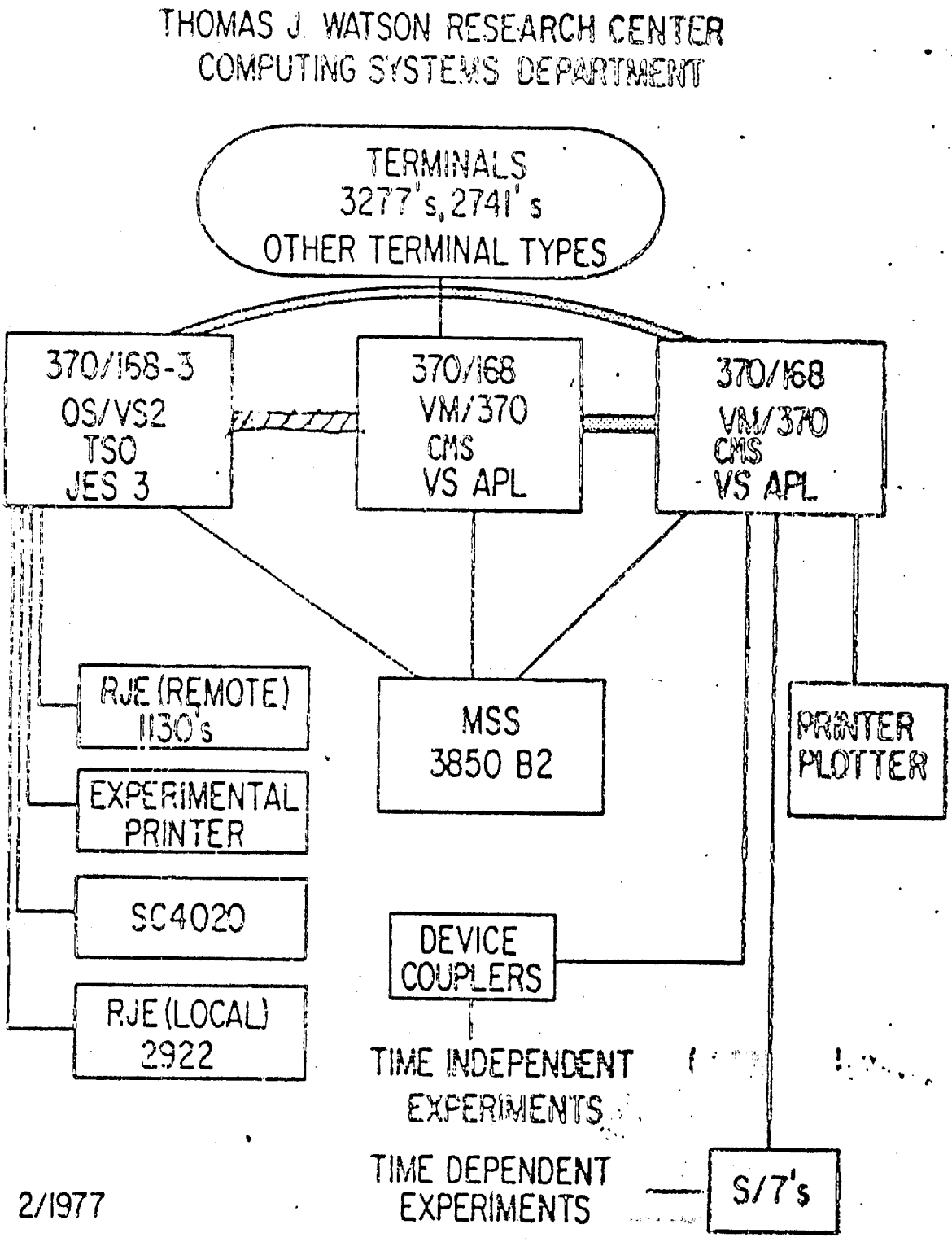
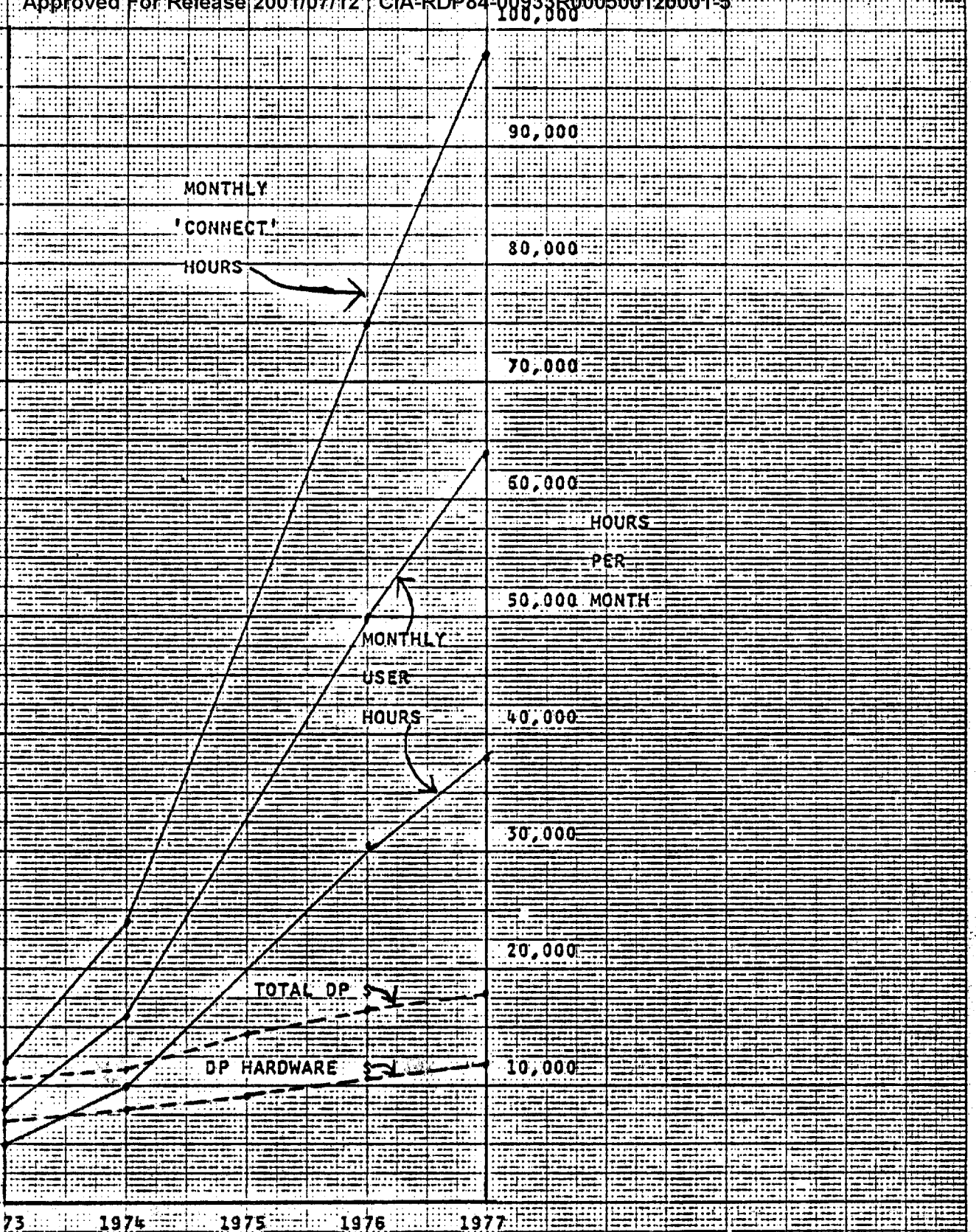


FIGURE 3



IBM T. J. WATSON COMPUTING CENTER

COMPUTING COSTS vs. USER COSTS

FIGURE 3A

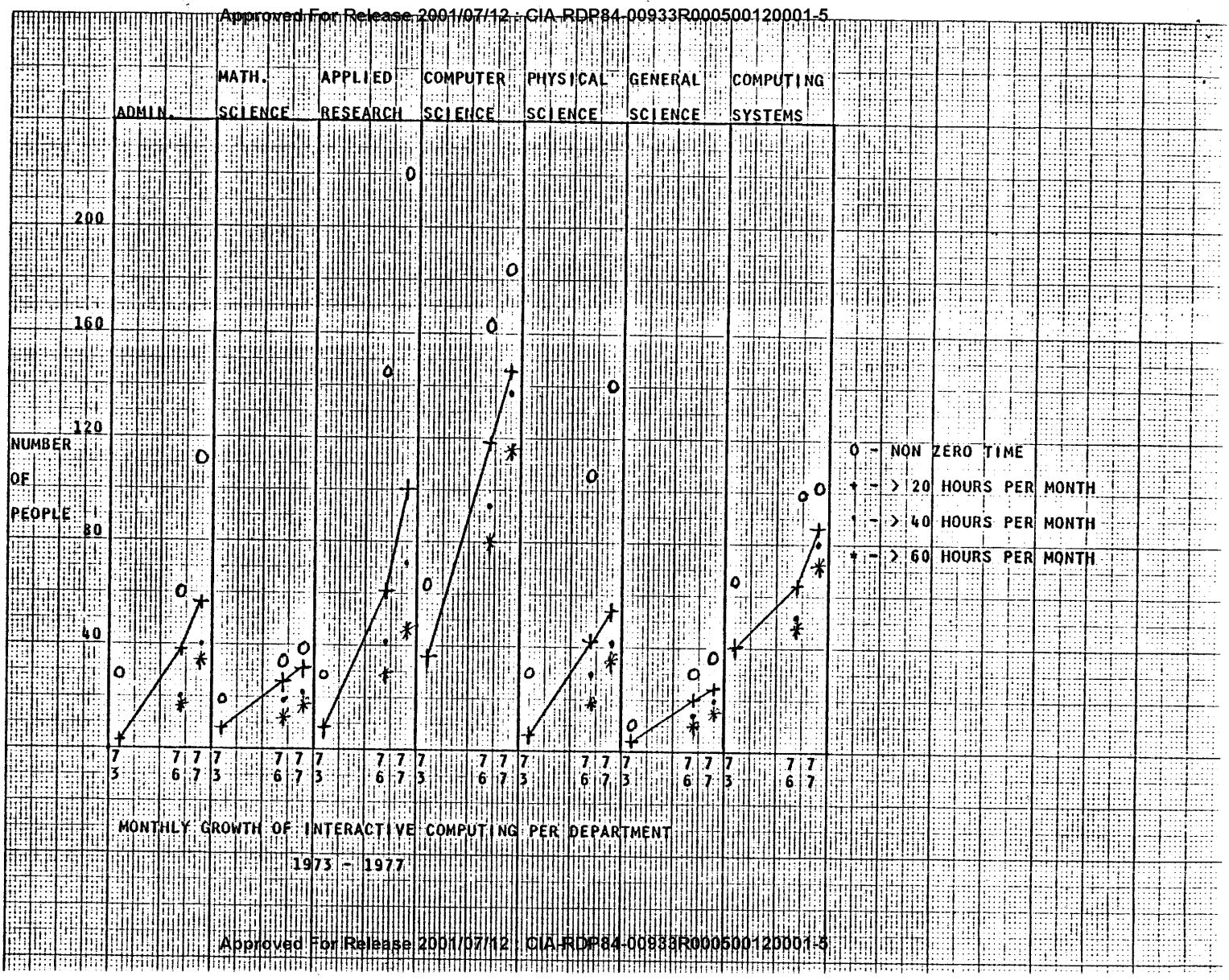


FIGURE 4

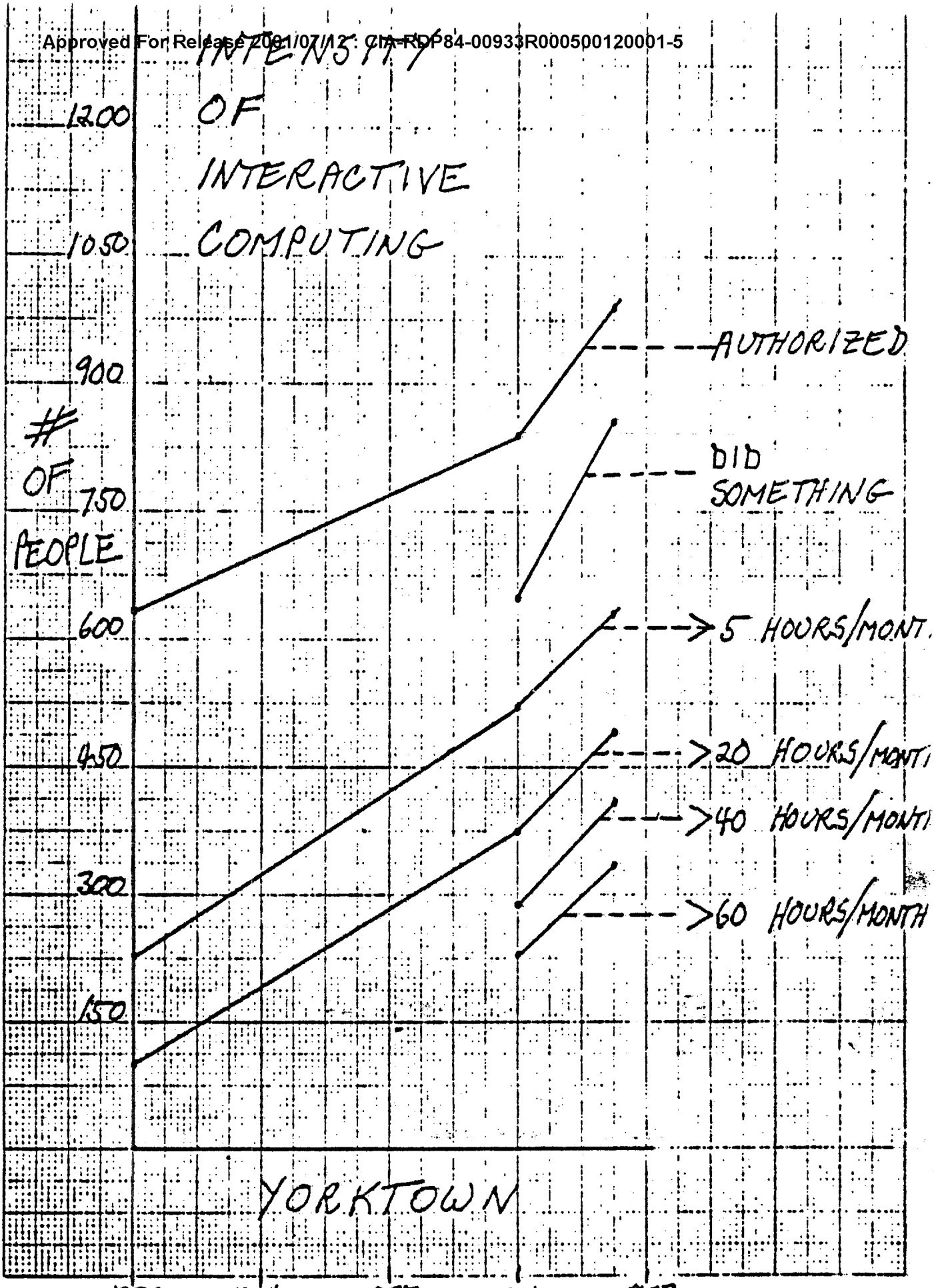


FIGURE 5

INTERACTIVE	BATCH
CONTINUOUS	DISCRETE
5 HOURS / DAY	1/4 HOUR / DAY
EVOLUTIONARY	STATIC
200 COMMANDS / DAY	9 STEPS / DAY
LOAD RANGE 100,000 : 1	LOAD RANGE 100,000 : 1

FIGURE 6

USER GROWTH

THE NUMBER OF COMMANDS EXECUTED PER PERSON DOUBLED IN FIVE MONTHS IN 1971 AT YORKTOWN

THE NUMBER OF COMMANDS EXECUTED PER COMMAND TYPED AT A TERMINAL IS 25 : 1 IN 1977 AT STANFORD

COMMAND PROCEDURES (30,000) OUTNUMBER PROGRAMS IN CONVENTIONAL LANGUAGES (15,000) AT YORKTOWN IN 1977

FIGURE 7

1	EXEC	6074	23.62226	182	99.78049
7	SCRIPT	2397	9.322132	119	72.56097
8	TEXT	1758	6.837008	114	69.51219
10	MODULE	728	2.831252	99	60.36584
6	ASSEMBLE	656	2.551238	73	44.51219
20	MAIL	261	1.01505	68	41.46341
45	PLI	1496	5.8256e7	64	39.02438
16	DEFAULTS	72	0.280014	58	35.36584
127	FORTRAN	723	2.811807	56	34.14633
19	$DIRECT	68	0.2644576	49	29.87804
52	MACLIB	84	0.326683	44	26.82925
22	VMAPLWS	350	1.361178	42	25.60974
95	MEMO	198	0.7700385	41	25.
15	ARCHIVES	56	0.2177886	40	24.39024
21	SYNONYM	40	0.1555633	40	24.39024
14	CONSOLE	52	0.2022323	38	23.17073
104	SYSIN	446	1.73453	36	21.95122
193	TERMTEXT	99	0.3850192	30	18.29268
53	TXTLIB	46	0.178876	30	18.29268
17	VSAPLWS	147	0.5716952	30	18.29268
148	JCL	253	0.983938	28	17.07317
277	PLIOPT	140	0.5676061	28	17.07317
23	OLDMAIL	27	0.1050052	25	15.2439
84	PLS2	272	1.05763	20	12.19512
226	DIAG	24	0.9333795E-01	16	9.756097
96	MACRO	177	0.6883677	12	7.317073
25	LISP	104	0.4046096	9	5.487804
451	PIC	40	0.1555633	9	5.487804
460	MVS	66	0.2644576	8	4.878048
112	OS	35	0.1361179	8	4.878048
144	PICTURE	35	0.1361179	6	4.878048
424	GRIN	56	0.2177886	6	3.658536
450	LABEL	39	0.1516742	6	3.658536
18	LISP370	76	0.2955703	6	3.658536
719	KGRAPH	85	0.3305721	4	2.439024
345	KSCRIPT	27	0.1050052	4	2.439024
499	INCLUDE	28	0.1088943	3	1.829268
110	SBS	305	1.18617	3	1.829268
94	BCPL	63	0.2450122	2	1.219512
723	CLIST	115	0.4472445	2	1.219512
743	PT05F001	94	0.3655738	2	1.219512
24	LISPLIB	31	0.1205615	2	1.219512
1668	RUNOFF	36	0.140007	2	1.219512
210	PISLIB	51	0.1983432	1	0.6097561
1557	FORT	34	0.1322288	1	0.6097561
285		7	0.2722358E-01		0.
223	SMONDANS	15	0.5833624E-01		0.
441	A	82	0.3189048		0.
123	ACCESS	7	0.2722358E-01		0.
119	AEL	33	0.1283397		0.
781	APAR	9	0.3500174E-01		0.
1669	AREADESC	52	0.2022323		0.
1670	AREADUMP	52	0.2022323		0.
565	ASM	43	0.1672305		0.
782	ASSEMBSV	25	0.9722704E-01		0.

FIGURE 8

EDITORS

75 PERCENT OF COMMANDS TYPED ARE FOR EDITING

MAJOR EDITING PROGRAMS HAVE HISTORICALLY BEEN NEGLECTED BY SYSTEM DEVELOPMENT GROUPS

MANY EDITORS DEVELOPED AND EVOLVED AT YORKTOWN

FIGURE 9

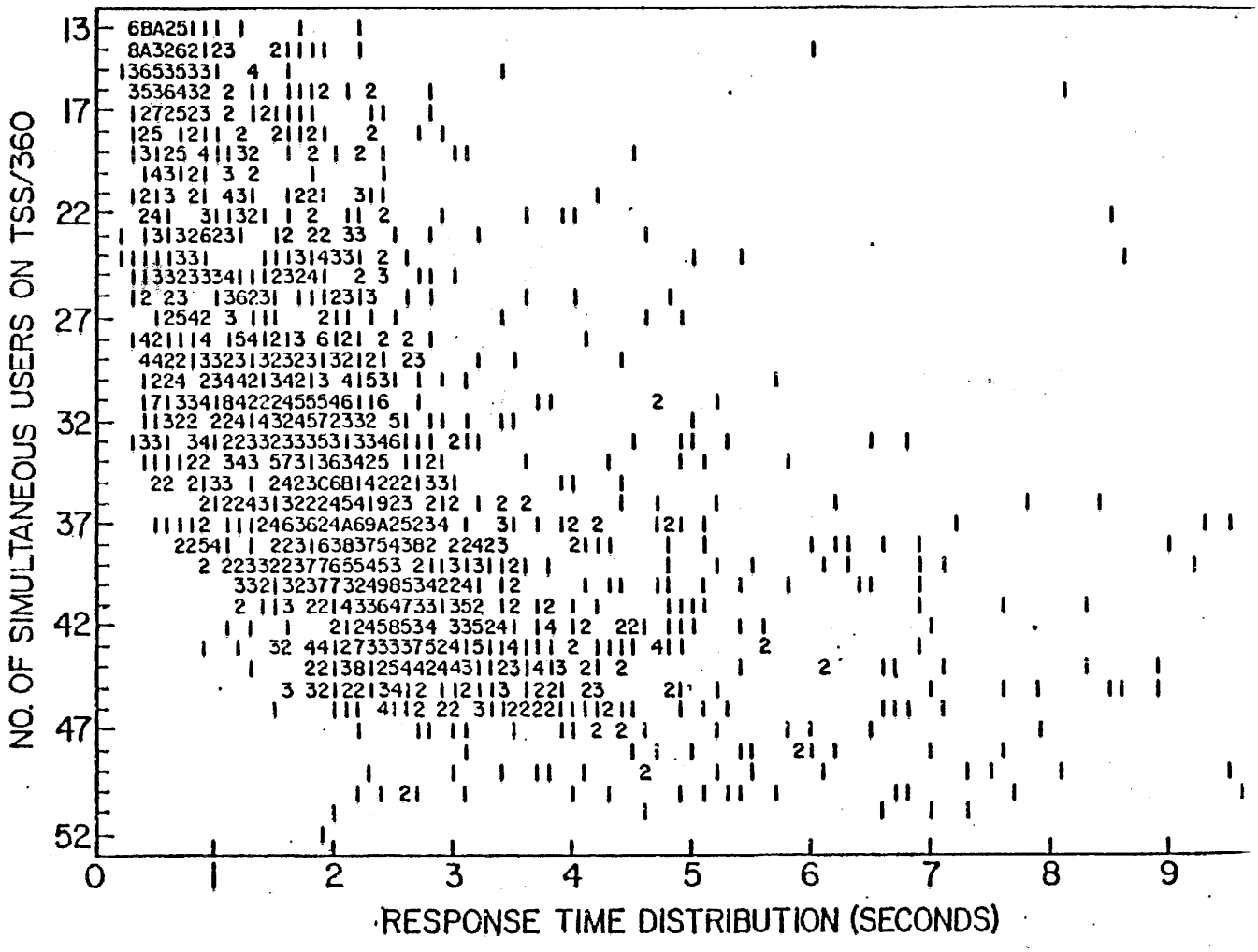


FIGURE 10

THE NUMBER OF SIMULTANEOUS SYSTEM USERS IS A MISLEADING MEASURE.

FIGURE 11

USER RESPONSE TIME IS DIRECTLY INFLUENCED

BY SYSTEM RESPONSE TIME (SRT)

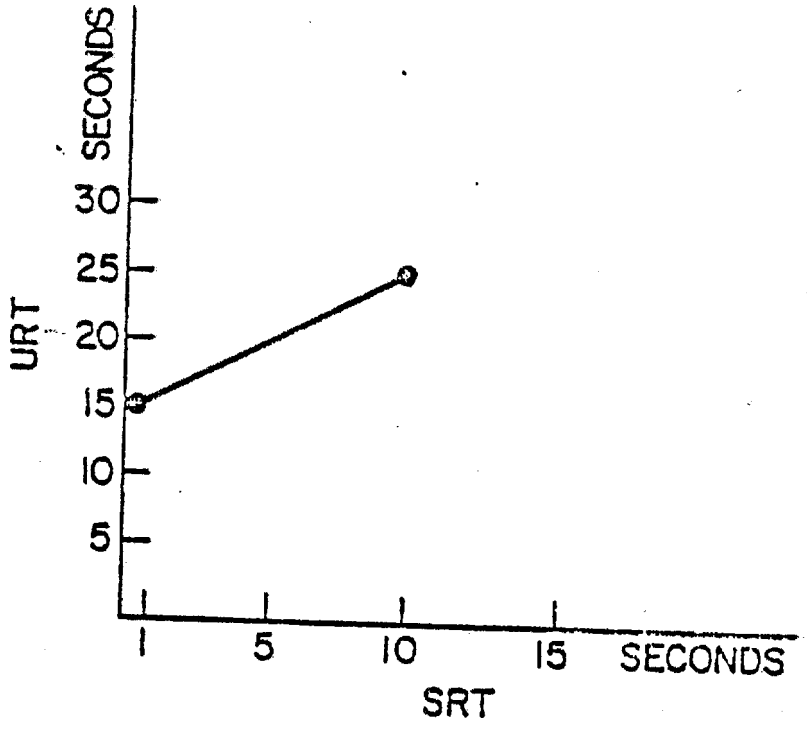


FIGURE 12

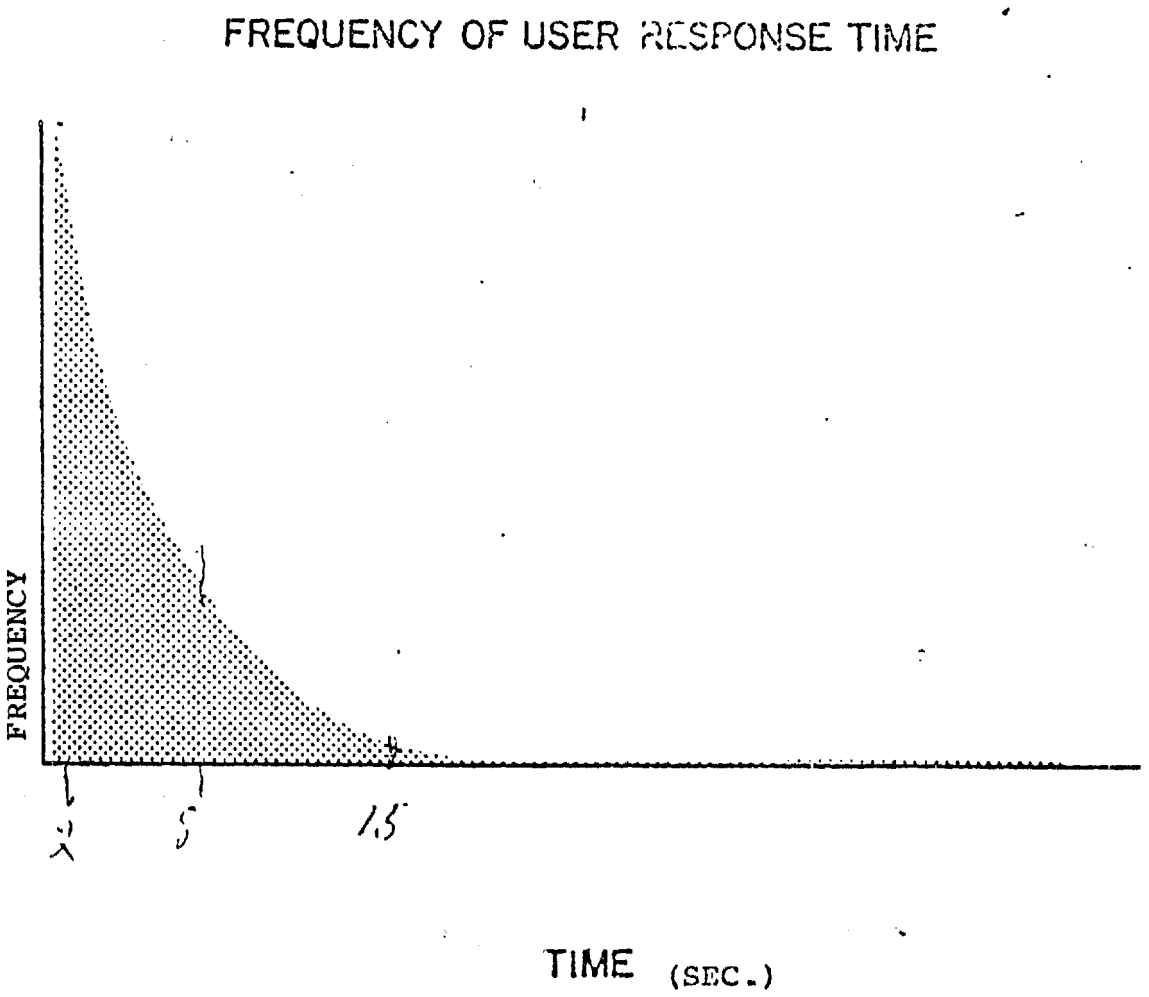


FIGURE 13

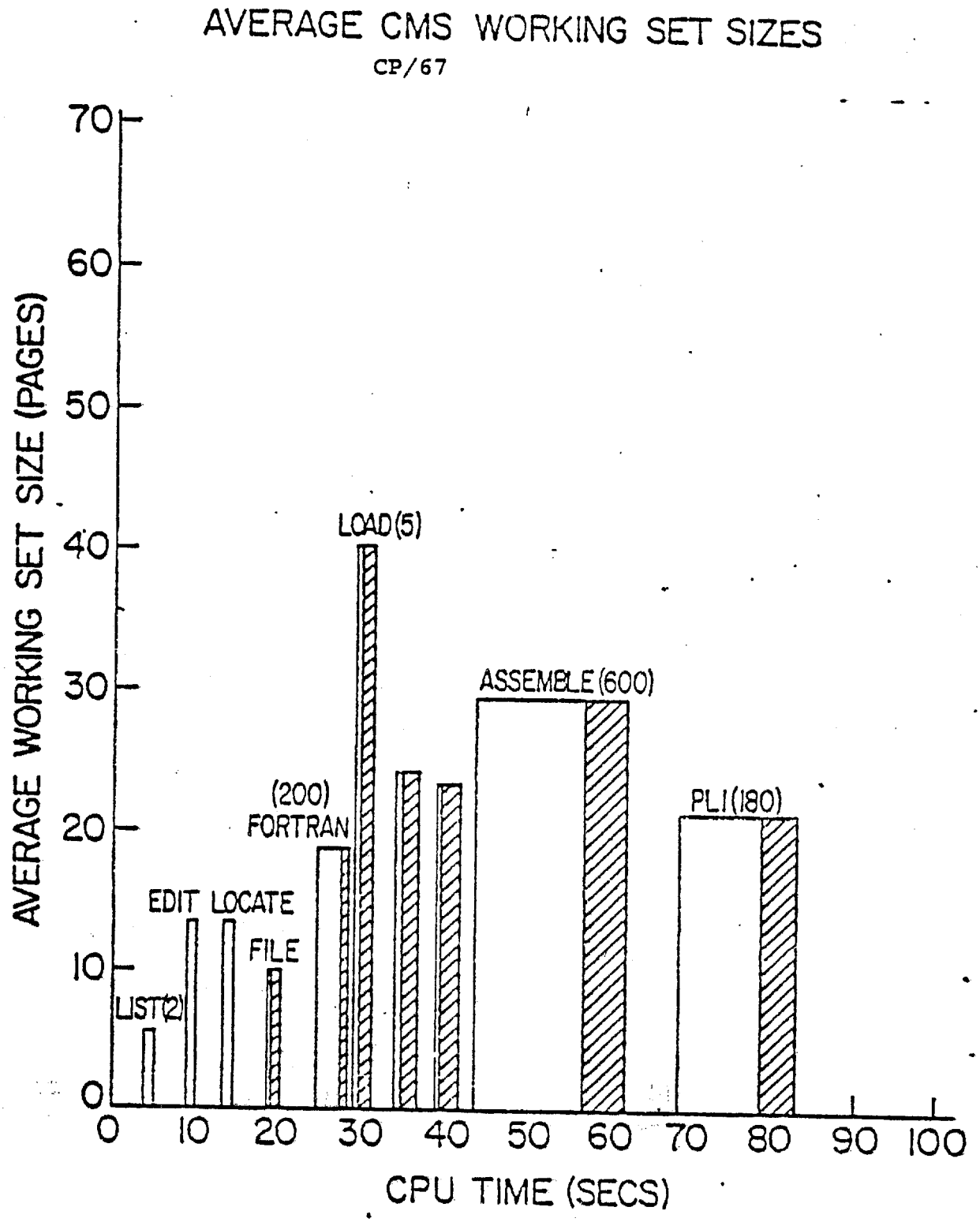


FIGURE 14

AVERAGE LISP WORKING SET SIZES

CP/67

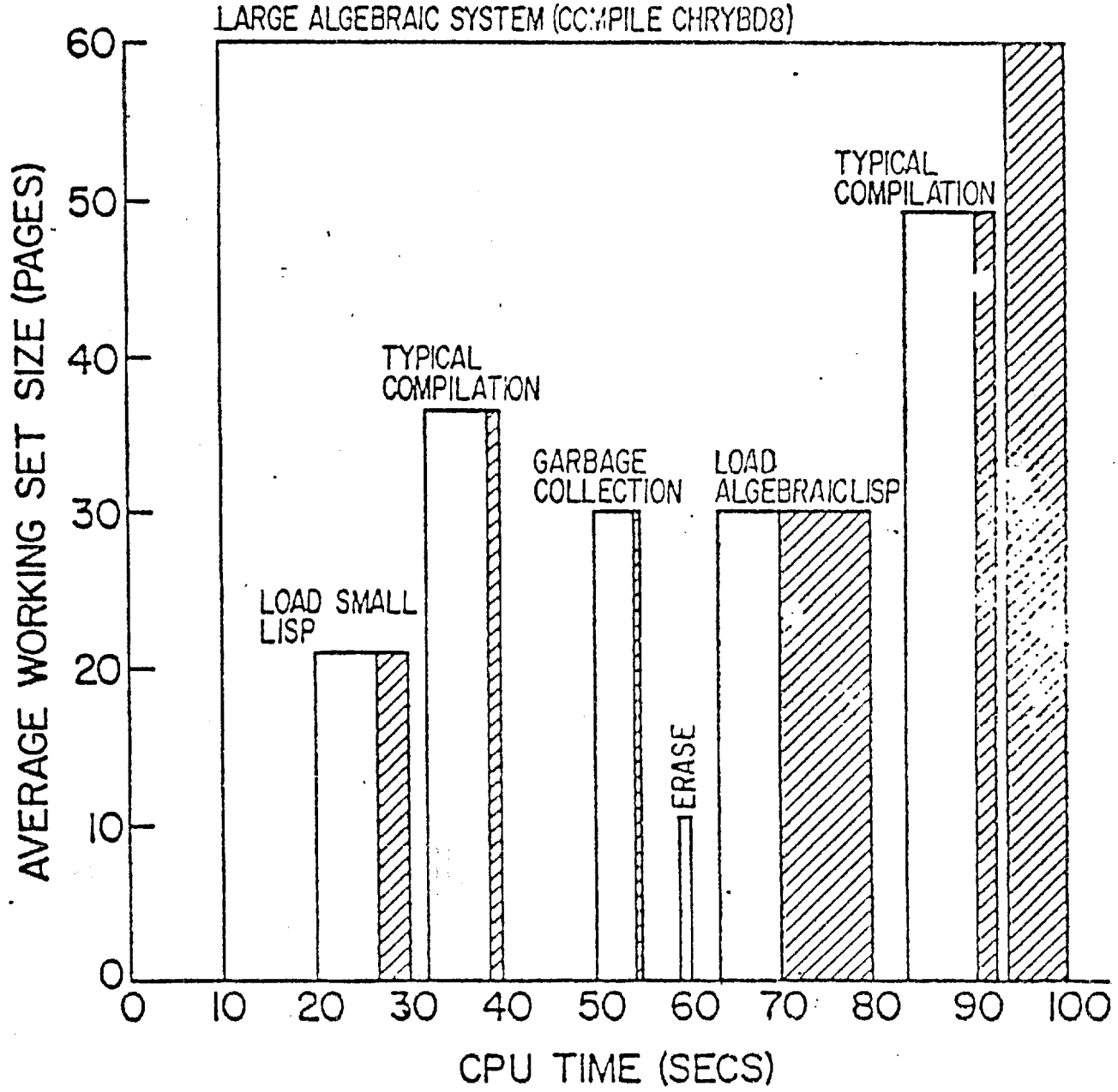


FIGURE 15

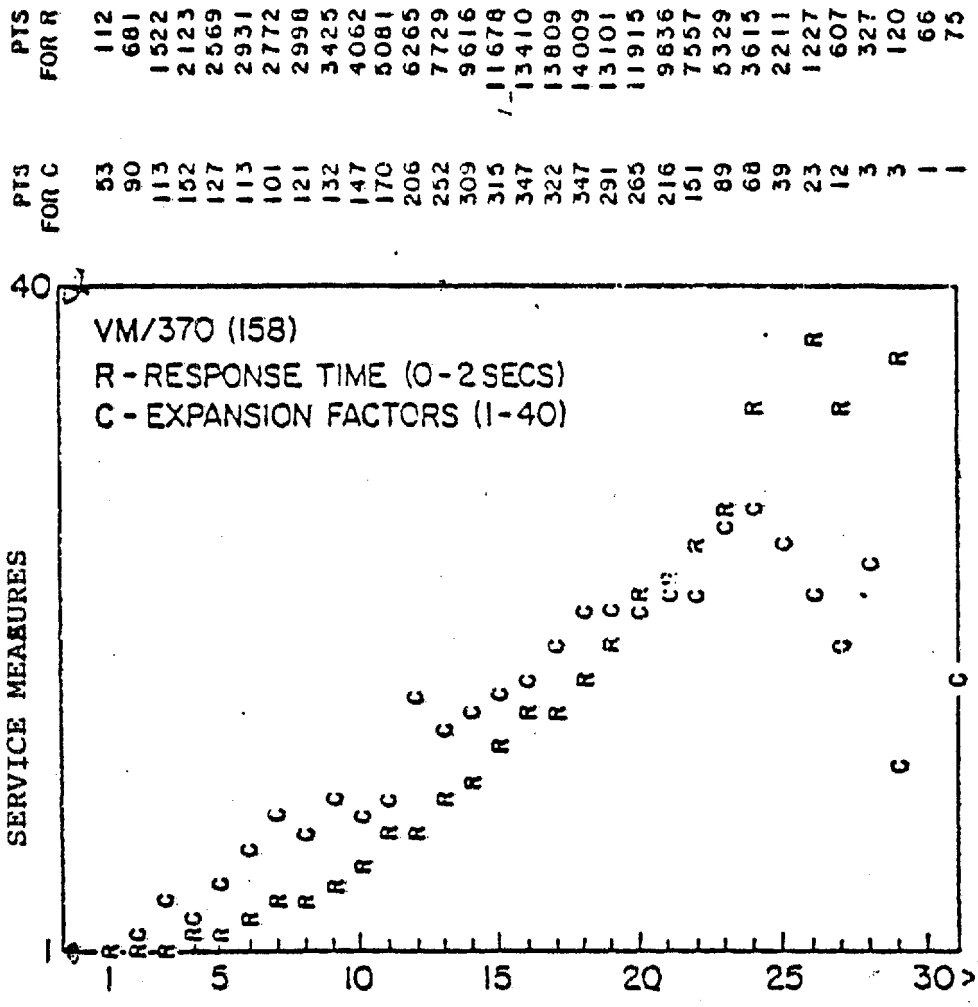


FIGURE 16

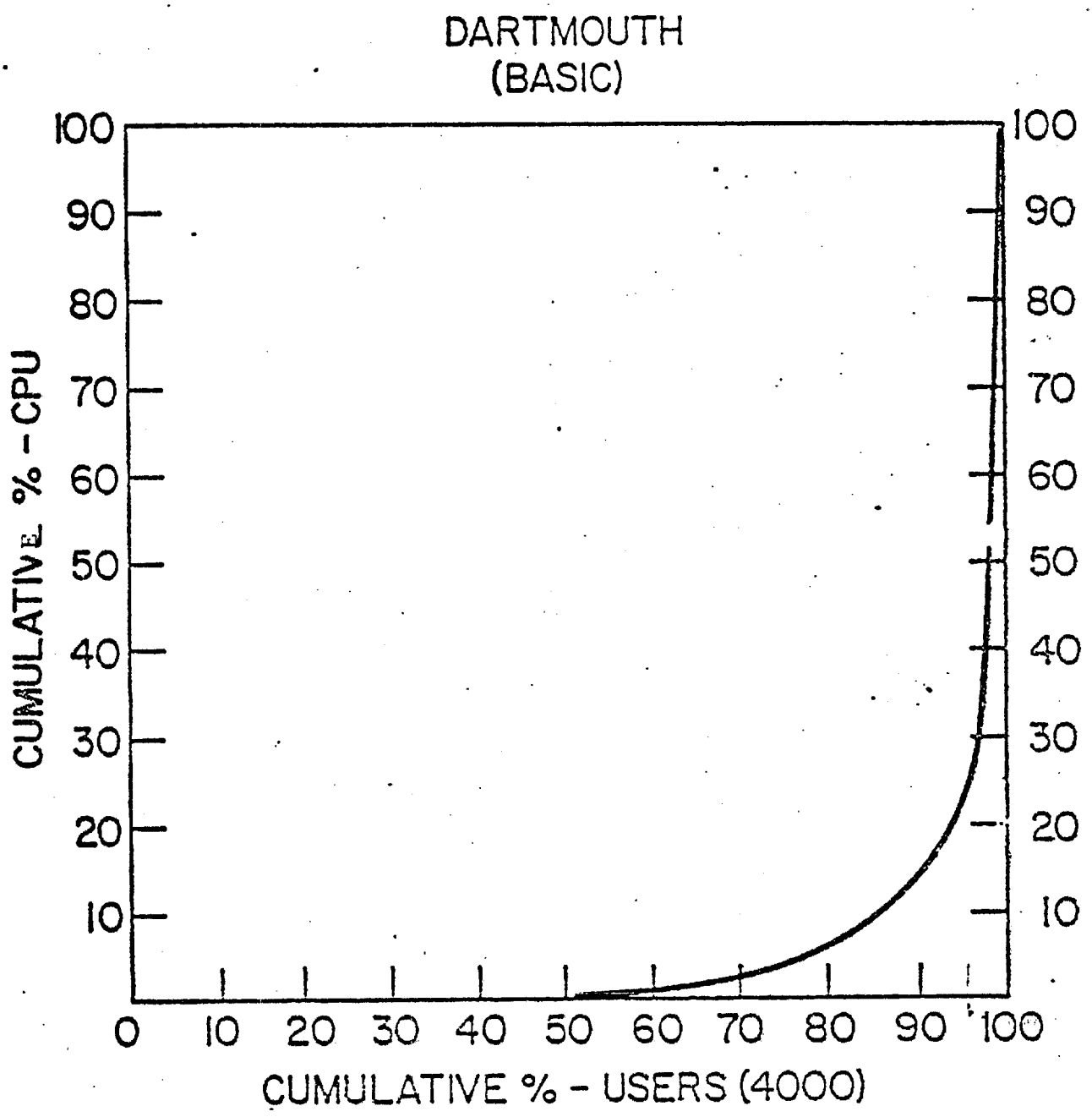


FIGURE 17

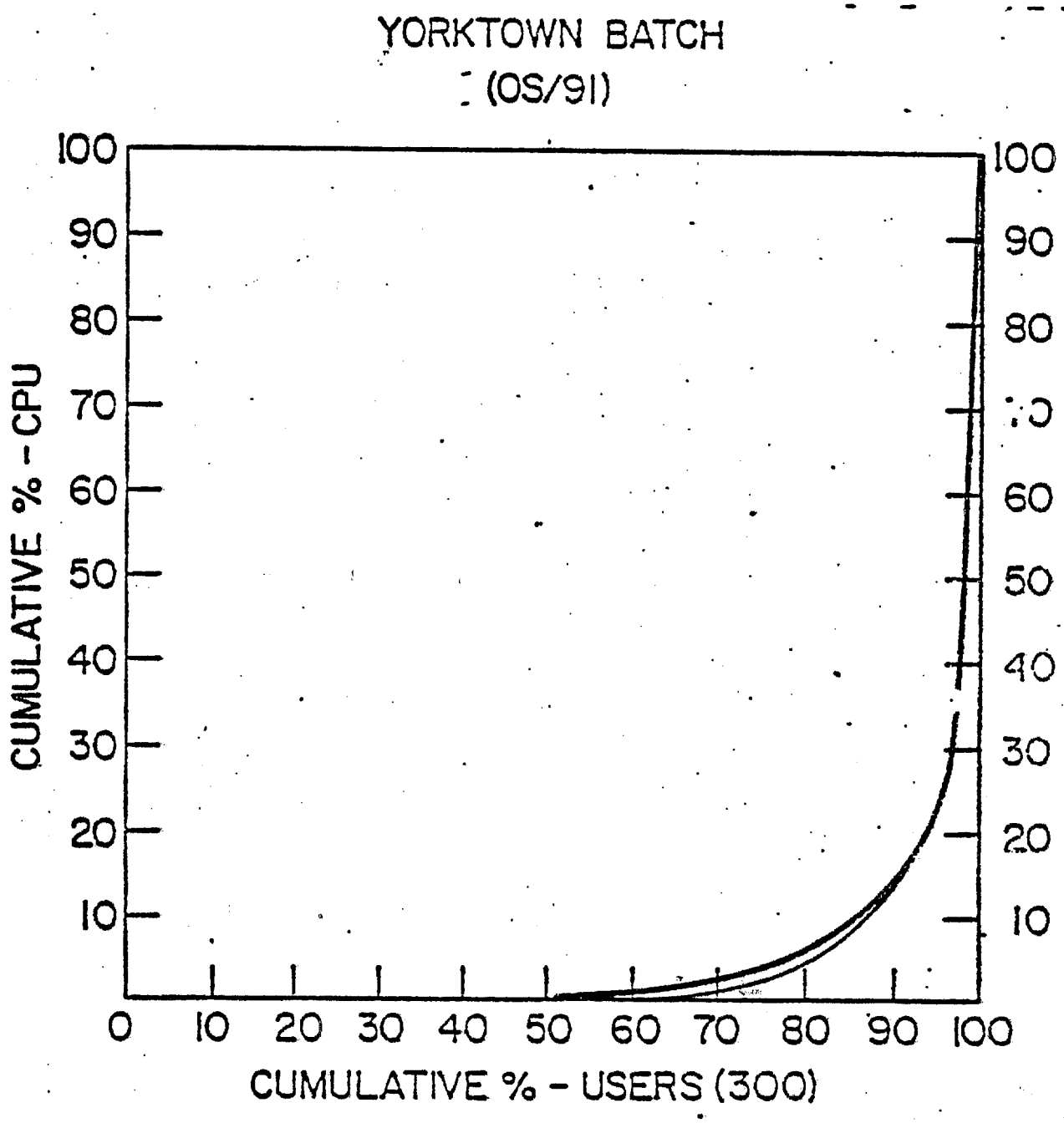


FIGURE 18

FEEDBACK TO USERS ENABLES INDIVIDUAL IMPROVEMENTS BY FACTORS OF 2 TO 20 IN SYSTEM USAGE.

FIGURE 18a

158 CAMBRIDGE BENCHMARKS (40 MIN.)

	WHEELER		RELEASE 3			
MEMORY	2 MEG	1 MEG	2 MEG		1 MEG	
RESPONSE	0.146	0.624	2.2	0.537	1.6	0.601
TRANSACTIONS	12,374	11,149	7,921	9,176	7,319	8,062
PROBLEM CPU	60.4%	50.3%	33.6%	37.1%	23.8%	43.5%
TOTAL CPU	98.5%	88.8%	66.6%	97.4%	51.2%	73.2%
DRUM I/O	95%	95%	65%	→		
DISK I/O	5%	5%	35%	→		
USERS	80	80	80	60	80	60

FIGURE 19

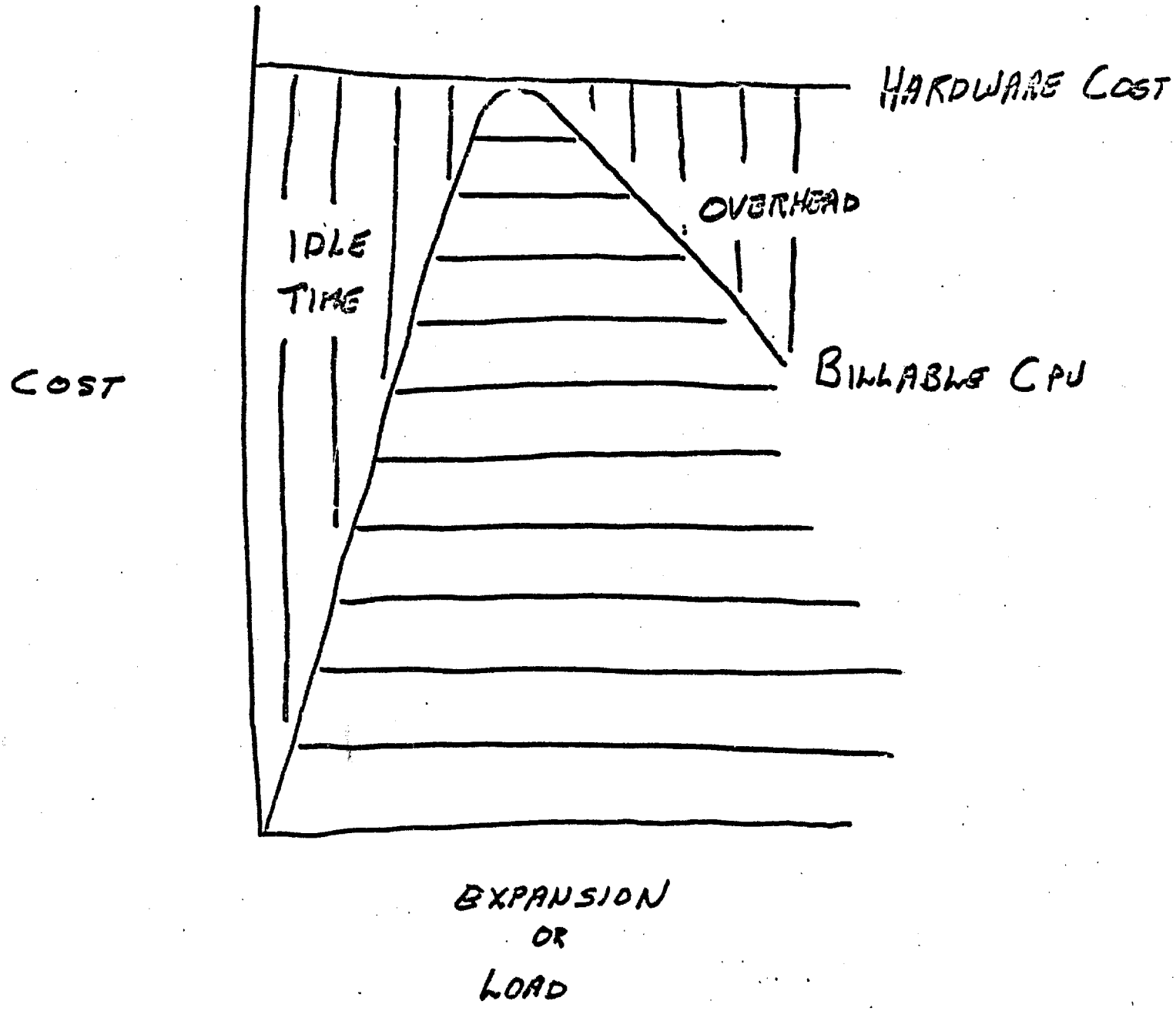


FIGURE 19A

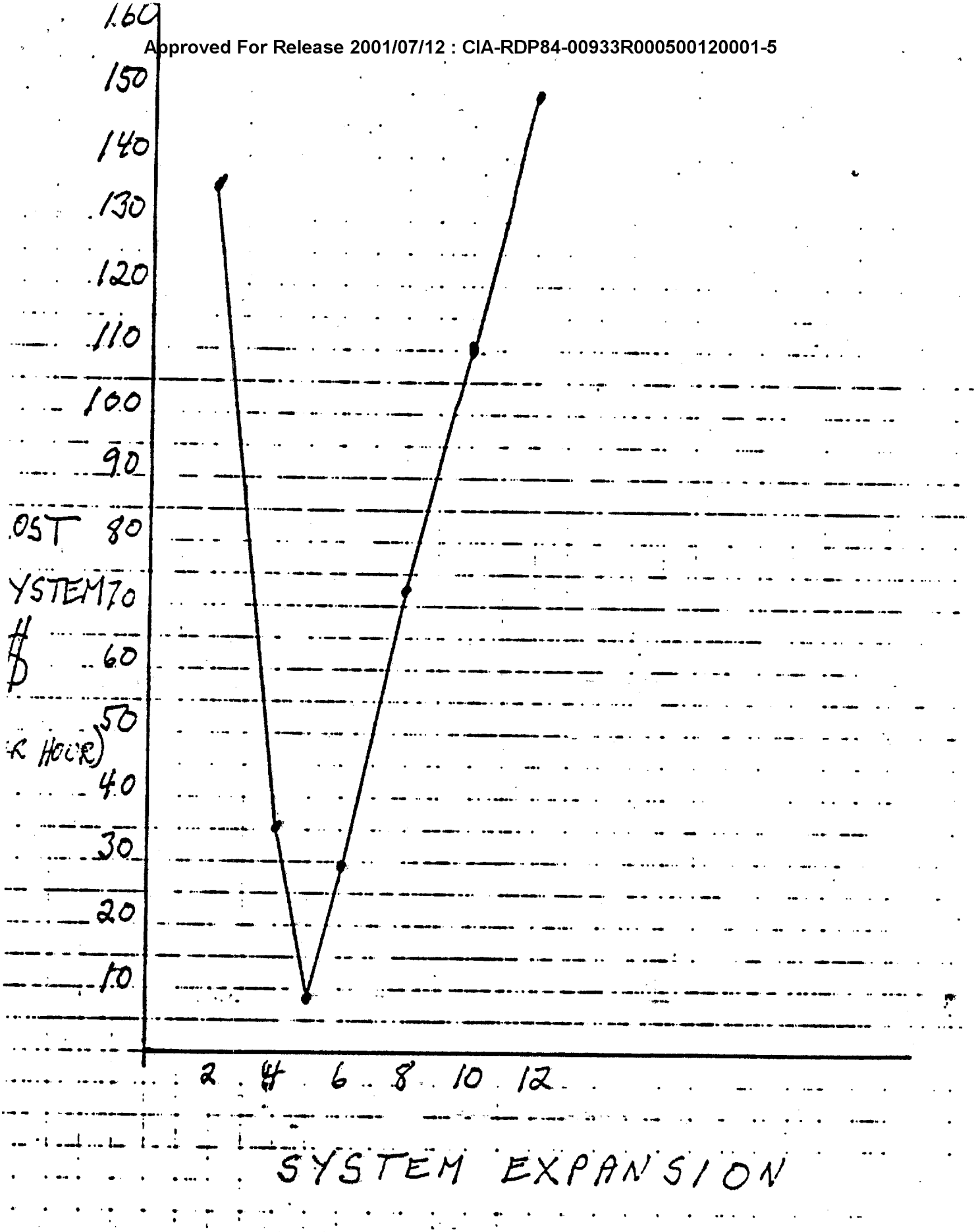


FIGURE 19B

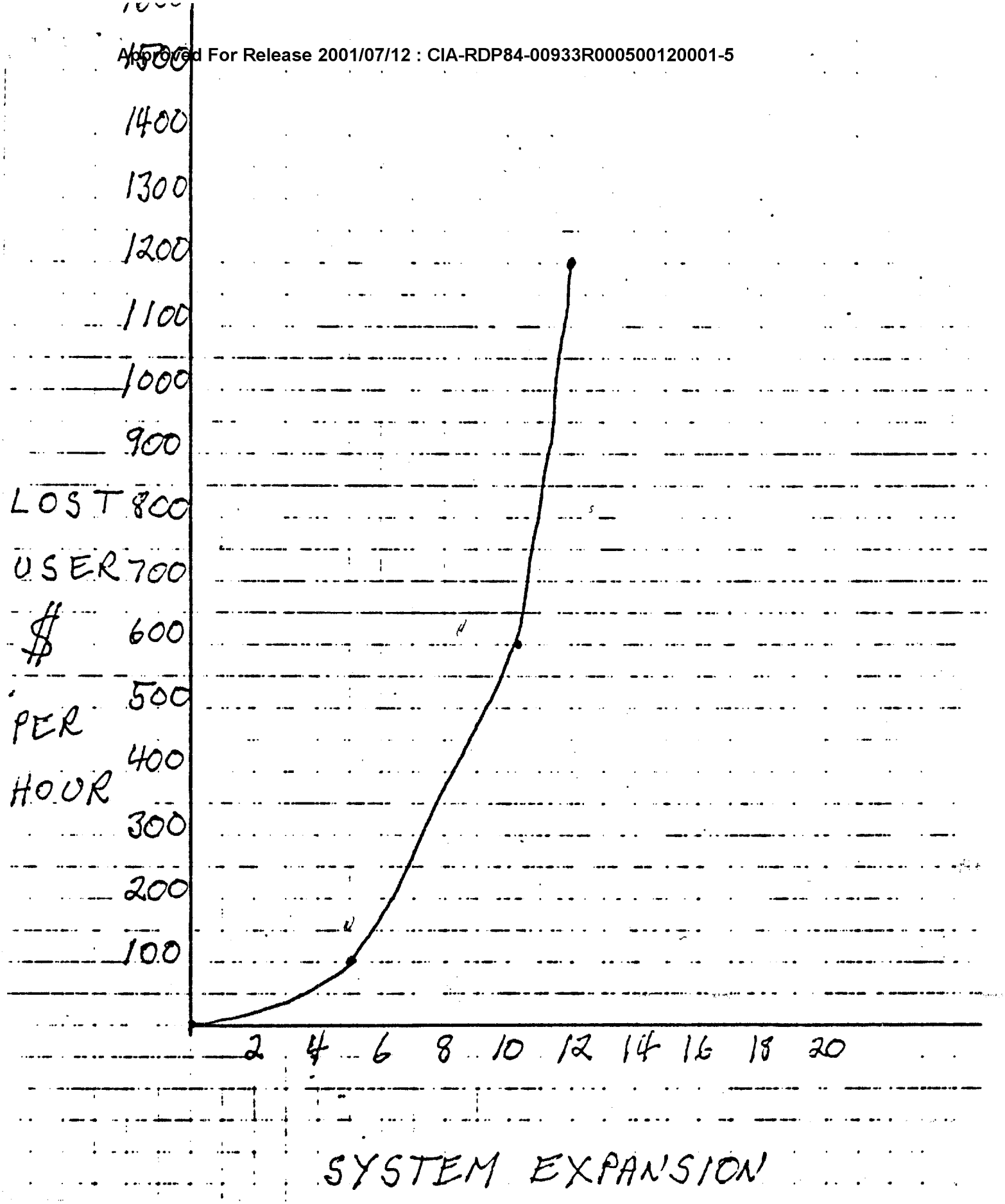


FIGURE 19C

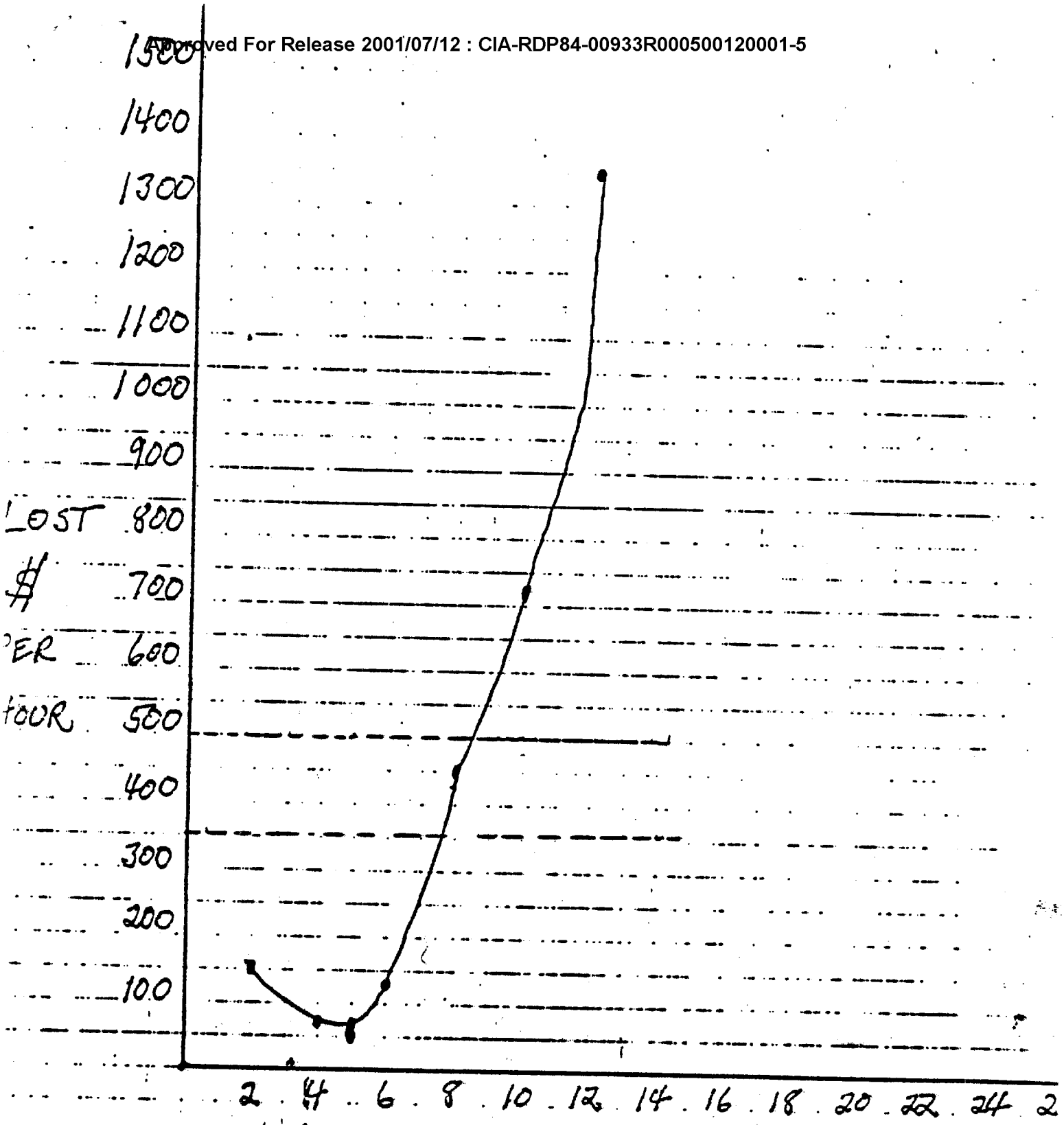


FIGURE 19D

STORAGE HIERARCHY

BOUNDARY PROBLEMS

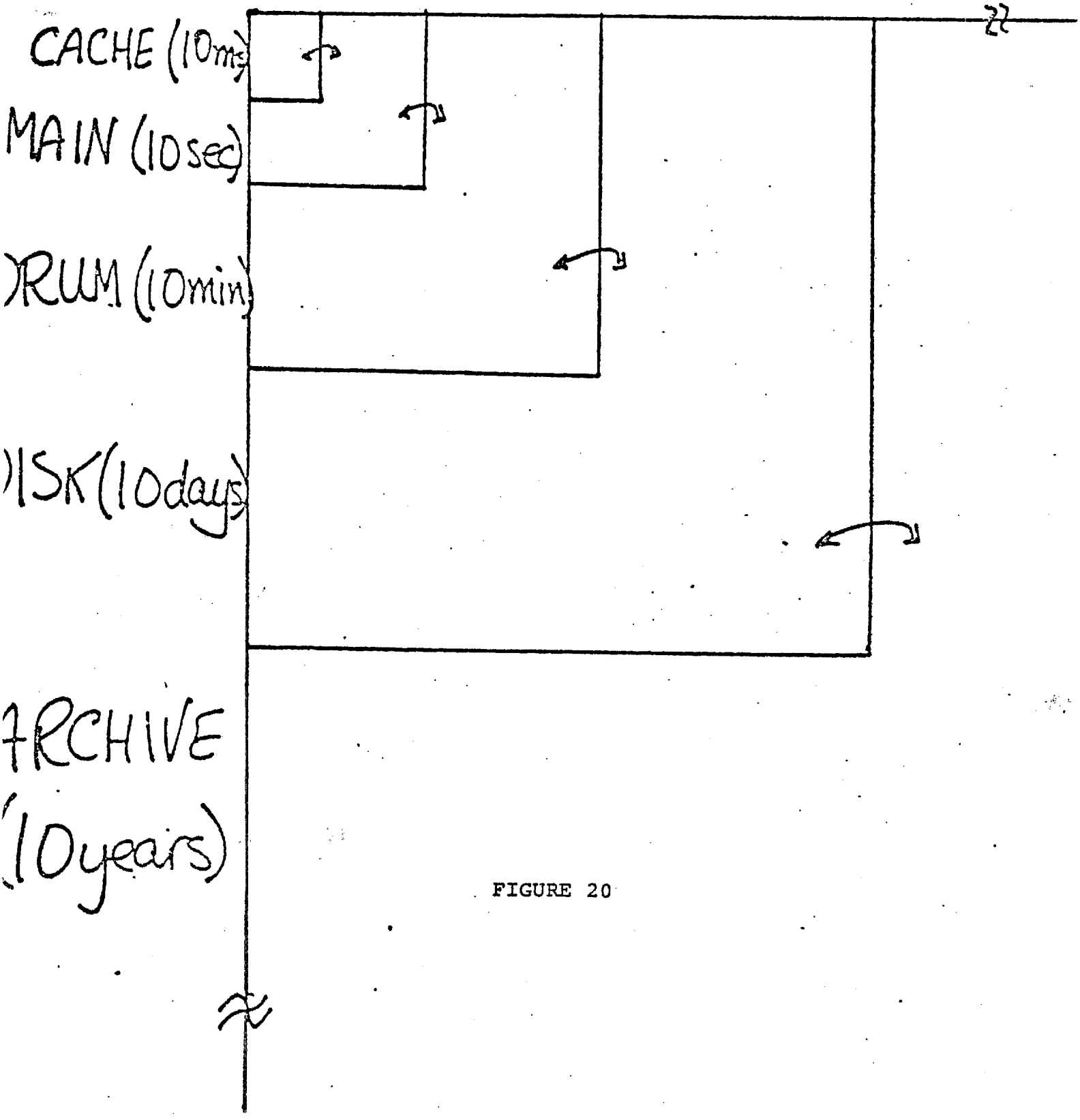


FIGURE 20

CPU SPEED

INTERACTIVE RESPONS

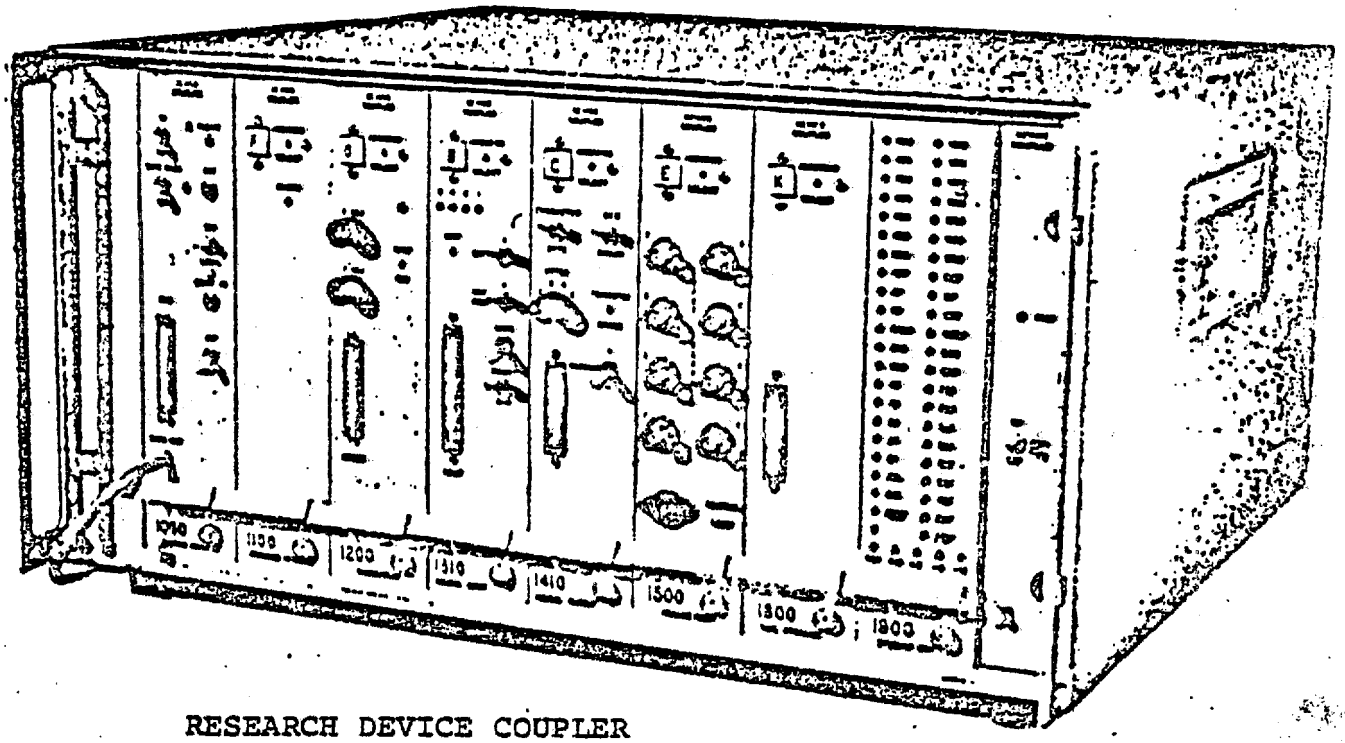
VIRTUAL MEMORY MANAGEMENT

USER GROWTH LIMITATION

FIGURE 21

LAB AUTOMATION	5100 - 7406 (DEVICE COUPLER)
LAB AUTOMATION	SYSTEM 7s TPVM
NETWORKING	VNET (RSCS)
NETWORKING	DATA STAGE
TEXT PROCESSING	TTF
INFORMATION	BULLETIN BOARD
INFORMATION	NATURAL LANGUAGE INQUIRY
MACHINE TOOLING	VS1 - APT
SPEECH FILING SYSTEM	SFS
DATA SPACE MANAGEMENT	MIGRATE - ARCHIVE
CMS BATCH	NEW BATCH MONITOR
DATA SPACE MANAGEMENT	CPDIR
LIBRARIES	19D - 19E - 19F

FIGURE 22



RESEARCH DEVICE COUPLER

FIGURE 23

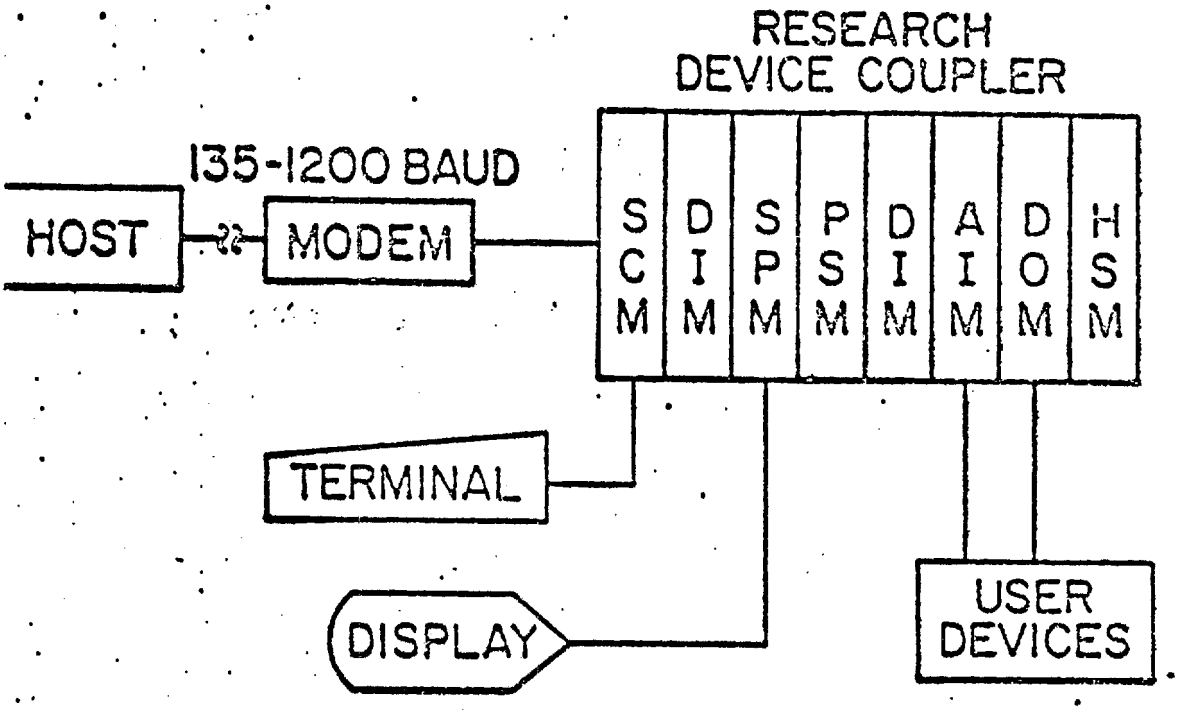


FIGURE 24

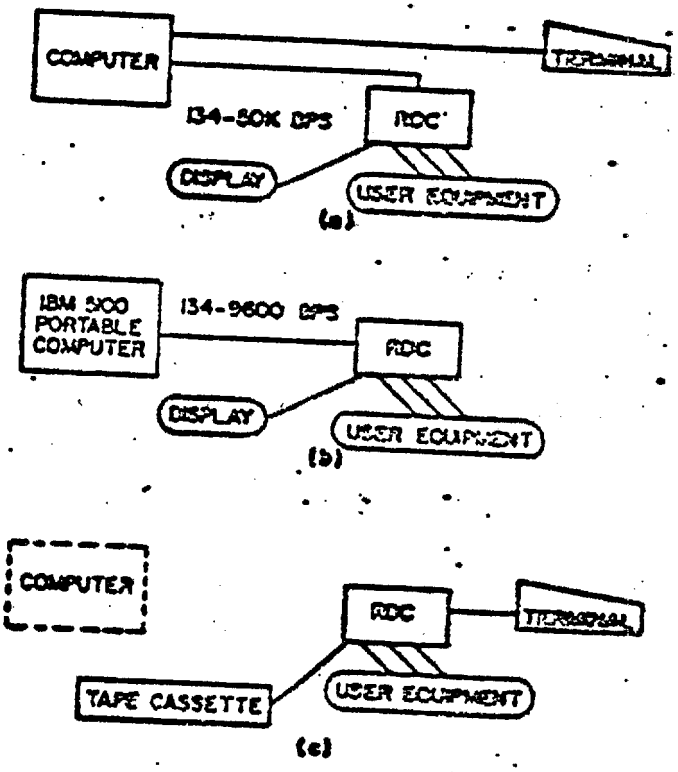


FIGURE 25

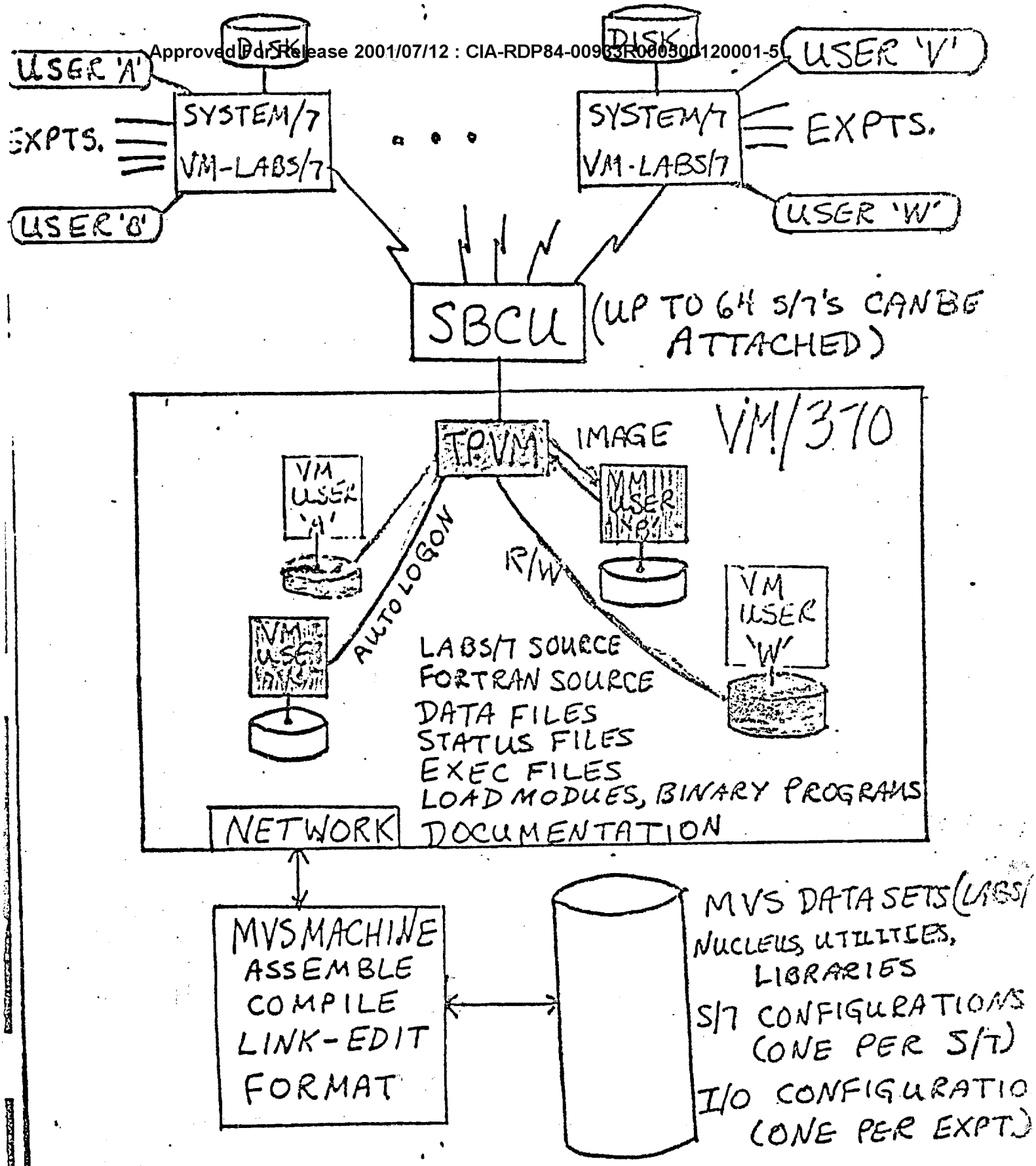
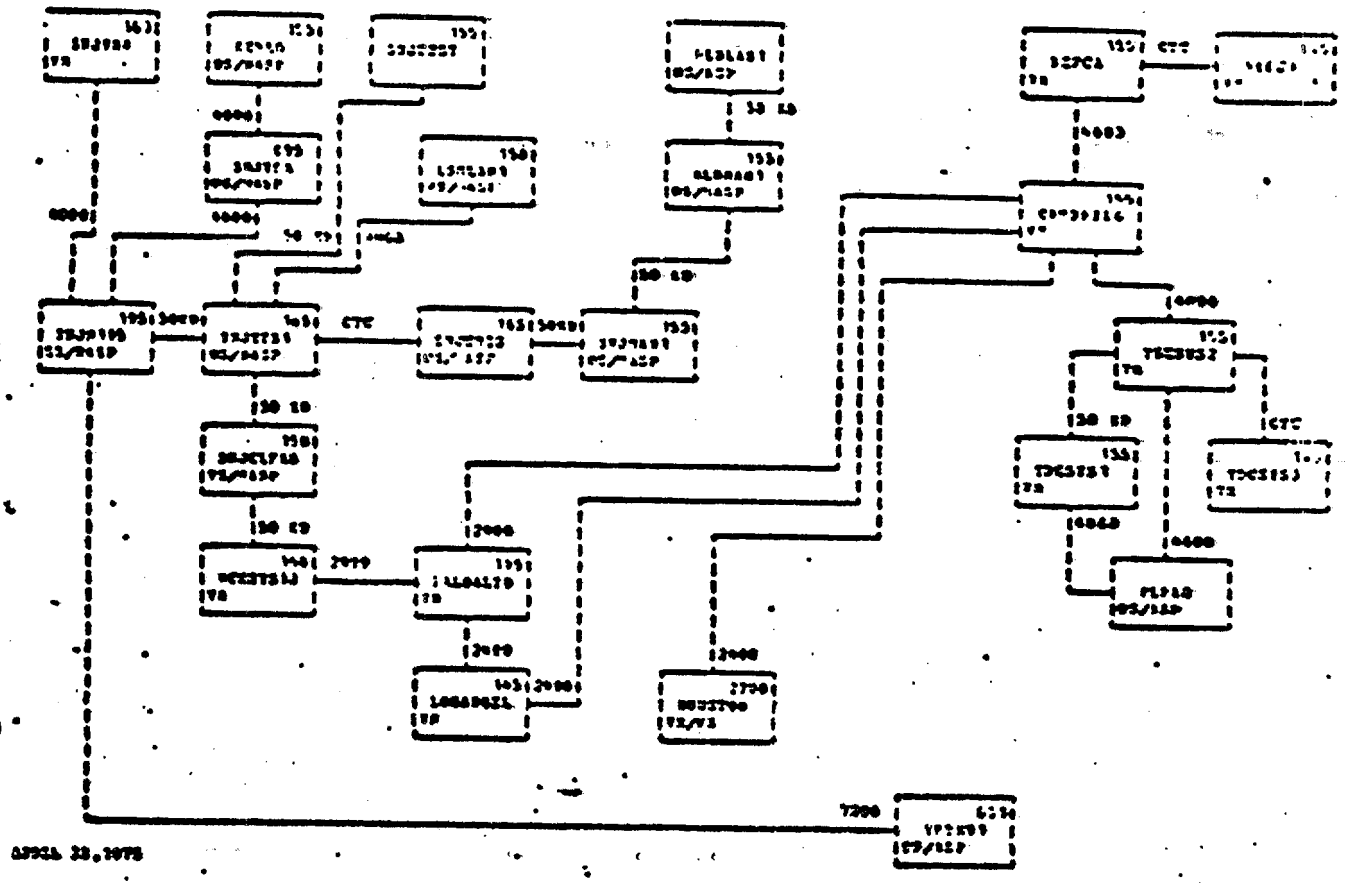
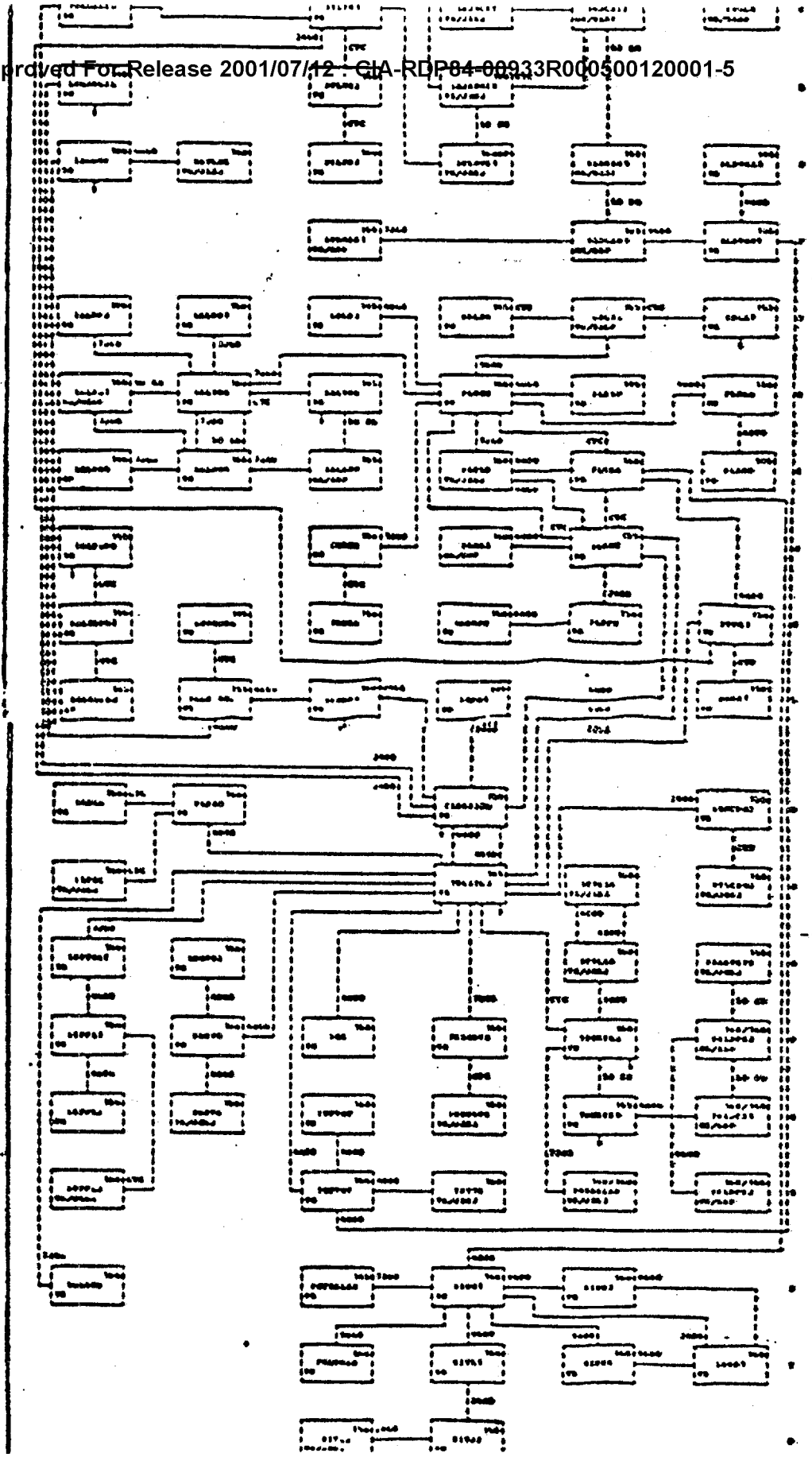


FIGURE 26



SUN NETWORK APRIL 1975

FIGURE 27



SUN NETWORK NOVEMBER 1976

FIGURE 28

	August	September	October	November	December	January
Total Monthly Job Counts						
Total	3054	3494	3854	3652	3724	4354
TTEXT	2454	2688	3025	3087	2981	3579
Non-TTEXT	600	806	829	565	743	775
First Shift	2568	2739	2926	2854	2999	3694
Normal Workday Job Counts						
Total	132	161	184	176	171	199
TTEXT	107	124	145	151	139	163
Non-TTEXT	26	37	39	26	32	36
First Shift	114	127	141	138	139	170
First Shift Small Jobs	95	102	112	113	112	134
First Shift Large Jobs	20	25	29	25	27	36
Percent Jobs From						
V	66	67	56	52	46	43
T	22	23	32	38	40	46
M	9	8	11	9	14	12
Percent Jobs Run On						
T	100	81	92	62	100	100
V	0	19	8	38	0	0
User Counts						
Total	152	185	190	193	175	186
TTEXT	128	150	154	153	146	156
Non-TTEXT	47	51	60	70	54	59
Normal Workday Total	40	48	54	56	52	59
Normal Workday TTEXT	35	41	45	50	44	52
Normal Workday Non-TTEXT	6	9	10	8	9	9
Monthly CPU Minutes						
Total	403	452	538	641	620	660
TTEXT	379	422	498	611	582	620
Non-TTEXT	24	31	39	29	38	40
First Shift	331	339	409	456	479	557
Normal Workday CPU Minutes						
Total	18	21	26	31	29	30
TTEXT	17	20	24	30	27	28
Non-TTEXT	1	1	2	1	2	2
First Shift	15	16	20	22	22	26
First Shift Small Jobs	9	9	11	12	12	12
First Shift Large Jobs	6	7	9	11	11	13
Turnaround Minutes						
First Shift Small Jobs	21	27	26	28	16	12
First Shift Large Jobs	22	26	30	54	36	23
DRAFT vs. FINAL						
Percent DRAFT (TTEXT only)	69	58	64	67	61	61
Percent FINAL (TTEXT only)	31	40	35	32	37	36
Total Monthly DRAFT	1688	1557	1922	2060	1826	2172
Total Monthly FINAL	1356	1880	1870	1551	1834	2063
Normal Workday DRAFT	73	71	91	101	84	97
Normal Workday FINAL	59	87	90	73	83	96
RTC vs. STC (TTEXT only)						
Percent RTC	73	68	64	52	49	47
Percent STC	24	28	32	43	47	49
LATE Option Usage						
Total	25	98	85	151	170	84
Normal Workday Percent	1	4	3	5	6	2